TIETO- JA SÄHKÖTEKNIIKAN TIEDEKUNTA

Jukka Pajukangas

REST-rajapinnan testauksen kattavuuden mittaaminen

Diplomityö
Tietotekniikan tutkinto-ohjelma
toukokuu 2020

Pajukangas J. (2020) REST-rajapinnan testauksen kattavuuden mittaaminen. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 94 s.

TIIVISTELMÄ

Internetin kautta käytettävien sovellusten kasvattaessa suosiotaan ja merkittävyyttään yhteiskunnassa on myös olemassa olevien web-rajapintojen määrä kasvanut. Representational State Transfer (REST) -ohjelmistoarkkitehtuurityyli on noussut suosituksi tavaksi toteuttaa web-rajapintoja. REST-pohjaisille rajapinnoille on kehitetty useita kuvauskieliä, joista suosituimmaksi on noussut OpenAPI-spesifikaatio (OAS). Ohjelmistotestaus nähdään usein todella kriittisenä osa-alueena ohjelmistokehitystä, joten myös REST-pohjaisten rajapintojen testaamiseen on kehitetty erilaisia testaustyökaluja.

Tässä työssä luotiin työkalu REST-pohjaisten rajapintojen testauksen kattavuuden mittaamiseen. API Specification Coverage tool (ASC) analysoi rajapinnan testauksen kattavuutta käyttämällä hyväkseen OAS:n mukaista rajapinnan kuvaustiedostoa ja testauksen aikana tallennettua verkkoliikennettä. ASC pystyy myös etsimään epäyhteneväisyyksiä rajapinnan kuvaustiedoston tapahtuneen verkkoliikenteen välillä, mikä voi paljastaa mahdollisia ongelmakohtia rajapinnan kuvauksessa, toteutuksessa tai testauksessa.

Työssä testattiin viittä erilaista OAS:n mukaisia rajapintoja automaattisesti testaavaa työkalua. Työkaluja testattiin rakennetun testijärjestelyn avulla, jossa työkalujen testisuorituksia valikoituja rajapintoja kohtaan evaluoitiin ASC:n avulla. Työkalujen tuottamaa testausta vertailtiin kattavuuden ja löytyneiden rajapinnan kuvauksen ja verkkoliikenteen epäyhteneväisyyksien määrän näkökulmasta. Testien tuloksien analysoinnissa havaittiin työkalun Schemathesis tuottavan parhaimpia tuloksia useimmissa tilanteissa.

Avainsanat: web-rajapinta, ohjelmistotestaus, ohjelmistotestaustyökalu, OpenAPI, OpenAPI-spesifikaatio.

Pajukangas J. (2020) Measuring test coverage of REST API. University of Oulu, Degree Programme in Computer Science and Engineering, 94 p.

ABSTRACT

With web-based applications having increased popularity and significance in society, also amount of existing web APIs (Application Programming Interface) has increased. Representational State Transfer (REST) software architectural style has become popular for creating web APIs. Several description languages for REST APIs have been developed, and most popular amongst them has been OpenAPI Specification (OAS). Software testing is often seen as critical part of the software testing, so there also exists different tools dedicated to REST API testing.

This thesis contributed a tool for REST API test coverage measurement. API Specification Coverage tool (ASC) analyses coverage of API testing by using OpenAPI document of API and traffic captured during testing effort. ASC is also able to search for inconsistencies between API description and traffic during test effort, which may uncover possible sticking points in API description, implementation or testing.

This work inspected five different tools dedicated to automated API testing of OpenAPI-conformant APIs. Tools were tested in test setup built in this thesis, which utilized ASC for evaluation of test runs made against selected set of test targets. Testing generated by tools were compared by examining results from viewpoint of coverage and amount of inconsistencies between test traffic and API description. Analysis of results revealed that tool named Schemathesis produced best results in most of the situations.

Keywords: web API, software testing, software testing tool, OpenAPI, OpenAPI Specification.

SISÄLLYSLUETTELO

ALKULAUSE

Tämä diplomityö tehtiin yhteistyössä Synopsyksen kanssa Security and Software Engineering Research Center (S2ERC) -tutkimuskeskuksen alla työskennellessäni Oulu University Secure Programming Group (OUSPG) -tutkimusryhmässä.

Haluan kiittää Synopsys Finland Oy:tä ja heidän työntekijöitään tuesta ja neuvoista tämän työn tekemisessä. Haluan erityisesti kiittää seuraavia henkilöitä: Pekka Pehkonen Synopsys Finland Oy:stä. Juha Röning (valvoja), Pekka Pietikäinen (tekninen ohjaaja) ja Teemu Leppänen (toinen tarkastaja) Oulun yliopistolta. Ilman heidän ohjaustaan, neuvojaan ja kärsivällisyyttään ei tämä diplomityö olisi ollut mahdollinen. Haluan myös kiittää kaikkia OUSPG:n työntekijöitä, ystäviäni ja perhettäni työtoveruudesta, vertaistuesta ja läsnäolosta tämän diplomityön tekemisen aikana.

Oulu, 3. kesäkuuta 2020

Jukka Pajukangas

LYHENTEIDEN JA MERKKIEN SELITYKSET

API	Application Programming Interface
ASC	API Specification Coverage tool
CI	Continuous Integration
HAR	HTTP Archive Format
HATEOAS	Hypermedia As The Engine Of Application State
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
ITU	International Telecommunication Union
JSON	JavaScript Object Notation
OAI	OpenAPI Initiative
OAS	OpenAPI Specification
OUSPG	Oulu University Secure Programming Group
REST	Representational State Transfer
S2ERC	Security and Software Engineering Research Center
SaaS	Software as a Service
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
YAML	YAML Ain't Markup Language

1. JOHDANTO

Internetin käyttö ja käyttäjämäärät ovat kasvaneet viime vuosikymmenien aikana: vuonna 2019 ITU:n (International Telecommunication Union) arvion mukaan yli puolet maailman väestöstä käytti internetiä [1]. Useimmat lukijat ovat varmasti myös havainneet, kuinka internet ja internetiä hyödyntävät palvelut ovat muuttaneet heidän arkipäivän elämäänsä esimerkiksi viimeisen kahdenkymmenen vuoden aikana niin työn kuin vapaa-ajan osalta. Ihmisten elämän ja yhteiskunnan toimintojen tukeutuessa kasvavassa määrin internetin kautta käytettäviin web-sovelluksiin on niiden sujuva kehittäminen ja toiminnan varmistaminen tärkeä kohdealue tutkimukselle ja liiketoiminnalle.

Web-rajapintoja julkaistiin vuonna 2019 kiihtyvällä tahdilla [2]. Web-rajapintojen kehityksessä suosituksi arkkitehtuuriksi on noussut REST-ohjelmistoarkkitehtuurityyli [3] [4]. REST-ohjelmistoarkkitehtuurityylillä saavutetaan etuja esimerkiksi skaalautuvuudessa, rajapintojen yleispätevyydessä ja komponenttien itsenäisyydessä [5 s. 105]. REST-pohjaisten rajapintojen kehitystyötä tukemaan on kehitetty useita rajapintojen kuvauskieliä. Kuvauskielien avulla voidaan luoda jäsentyneitä kuvauksia rajapinnoista ja siten helpottaa ja nopeuttaa rajapinnan kehitysprosessia [6] [7] [8]. Kuvauskielistä suosituimmaksi on noussut OpenAPI-spesifikaatio [4] [9].

Ohjelmistotestaus on kriittinen osa-alue ohjelmistokehityksessä. Ohjelmistotestauksen tavoitteena on löytää ohjelmistoista virheitä, jotta ne voitaisiin korjata ennen niiden realisoitumista todellisiksi haittavaikutuksiksi. Ohjelmistojen virheettömästä toiminnasta riippuvien yhteiskunnan osa-alueiden lisääntyessä myös ohjelmistovirheiden seurauksien vaikutukset kasvavat. REST-pohjaisten web-rajapintojen testaukseen on kehitetty erilaisia työkaluja, jotka tarjoavat mahdollisuuksia erilaisen testauksen suorittamiseen. REST-pohjaisten web-rajapintojen testauksen evaluointi on jäänyt vähäisemmän huomion kohteeksi kuin testauksen tekeminen itsessään, joten mahdollisuuksia testauksen evaluoinnin aputyökalujen kehitykselle on olemassa.

Tässä työssä esitellään työkalu OpenAPI-spesifikaation mukaisen rajapinnan testauksen evaluointiin. Työkalu käyttää testauksen kattavuuden laskemiseen rajapinnan kuvaustiedostoa ja testauksen aikana tallennettua verkkoliikennettä. Lisäksi työkalu suorittaa verkkoliikenteen analysoinnin, joka etsii testien aikana ilmenneet verkkoliikenteen epävastaavuudet suhteessa rajapinnan kuvaustiedostoon. Työkalua käytettiin osana testijärjestelyä, jossa evaluoitiin viiden erilaisen automaattisesti testausta generoivan työkalun tuottamaa testausta. Saadut tulokset esitetään ja analysoidaan työn niille varatuissa osioissa.

2. TAUSTA

REST (Representational state transfer) -ohjelmistoarkkitehtuurityyli on Roy Fieldingin esittelemä yleinen ohjelmistoarkkitehtuurityyli. Noudattamalla REST-ohjelmistoarkkitehtuurityylin periaatteita, järjestelmän on mahdollista parantaa skaalautuvuutta, rajapintojen yleispätevyyttä ja komponenttien itsenäisyyttä [5 s. 105].

REST-ohjelmistoarkkitehtuurityyliä hyödyntävät web-rajapinnat ovat nousseet suosituksi kehittäjien ja käyttäjien keskuudessa [3] [4]. REST-pohjaisille rajapinnoille on kehitetty useita kuvauskieliä, joiden tarkoitus on tarjota jäsentynyt ja helposti sekä kone- että ihmisluettava kuvaus rajapinnasta, mikä tuo etuja rajapinnan kehitystyöhön [8].

HTTP (Hypertext Transfer Protocol), eli hypertekstin siirtoprotokolla, on sovelluskerroksen tiedonsiirtoon tarkoitettu protokolla hypermediajärjestelmille. HTTP:n tyypillinen käyttötapaus on verkkosivun tai binääridatan, kuten kuvien, siirtäminen palvelimelta asiakasohjelmalle [10]. REST-ohjelmistoarkkitehtuurityyli on protokollariippumaton, mutta HTTP on yleinen tapa käyttää REST-pohjaisia rajapintoja.

Ohjelmistotestaus on kriittinen osa-alue ohjelmistokehityksessä. REST-pohjaisten rajapintojen testaukseen on luotu työkaluja erilaisiin tarpeisiin, mutta rajapintoihin kohdistuvan testauksen evaluoinnissa vaikuttaa olevan tilaa kehitykselle.

2.1. REST-ohjelmistoarkkitehtuurityyli

REST-ohjelmistoarkkitehtuurityyli on yleinen ohjelmistoarkkitehtuurityyli, jota voidaan hyödyntää web-rajapintojen kehityksessä. REST-ohjelmistoarkkitehtuurityylissä on kuusi määriteltyä rajoitetta [5 s. 76-86]. Näistä rajoitteista viisi rajoitetta (asiakas-palvelin, tilattomuus, välimuistin käyttö, yhdenmukainen rajapinta, kerroksittainen järjestelmä) ovat pakollisia ja yksi vapaaehtoinen (ladattava koodi). Jos järjestelmä ei noudata jotakin rajoitetta näistä viidestä pakollisesta rajoitteesta, järjestelmän ei voida sanoa olevan REST-pohjainen. REST-ohjelmistoarkkitehtuurityyliä noudattavasta järjestelmästä käytetään myös termiä “RESTful”. Järjestelmän noudattaessa näitä rajoitteita on mahdollista saavuttaa etuja sen skaalautuvuudessa, rajapintojen yleispätevyydessä, komponenttien itsenäisyydessä, vuorovaikutuksen latenssin vähentämisessä välikerrosten komponenttien avulla, järjestelmän turvallisuudessa sekä vanhentuneiden järjestelmän osien kapseloinnissa [5 s. 105].

2.1.1. Asiakas-palvelin -rajoite

Ensimmäinen pakollinen rajoite on asiakas-palvelin [5 s. 78]. Asiakas-palvelin -rajoitteen perusajatus on jakaa järjestelmä kahteen eri osapuoleen, joilla on erilainen toiminnallisuus ja vastuu järjestelmän toiminnasta. Palvelimen tehtävä on vastata asiakkaan lähettämiin pyyntöihin ja toimia niiden mukaan. Asiakkaan tehtävä on lähettää palvelimelle pyyntöjä oikeassa muodossa ja käsitellä vastaukset tarpeellisella tavalla. Palvelin ja asiakas ovat toisistaan riippumattomia, mikä

mahdollistaa niiden kehittämisen täysin erillään. Palvelimen ja asiakkaan sisäisellä toiminnalla ei ole merkitystä järjestelmän toiminnan kannalta, kunhan molemmat noudattavat viestinnässään samaa rajapintaa. Esimerkkinä tällaisesta järjestelmästä voisi toimia sähköpostipalvelin, joka hoitaa viestien lähettämisen, vastaanottamisen ja tallentamisen, sekä sähköpostin asiakasohjelma, joka tarjoaa graafisen käyttöliittymän sähköpostipalvelimen tarjoamiin palveluihin.

2.1.2. Tilattomuus-rajoite

Toinen pakollinen rajoite on tilattomuus [5 s. 78-79]. Tilattomuus-rajoite liittyy läheisesti asiakas-palvelin-rajoitteeseen. Tilattomuus-rajoite vaatii, että palvelin ei säilö asiakkaansa istunnon tilaa palvelimelle. Jokaista asiakkaan tekemää kyselyä palvelimelle kohdellaan omana kokonaisuutenaan ja riippumattomana kaikista muista kyselyistä. Jokaisen asiakkaan lähettämän kyselyn on sisällettävä kaikki palvelimen tarvitsemat tiedot, jotta kyselyn suorittaminen olisi mahdollista. Näiden seikkojen vuoksi on asiakkaan säilöttävä oman istuntonsa tila itse. Tilattomuus-rajoitteen noudattaminen tekee järjestelmän luotettavammaksi, skaalautuvammaksi ja helpommin monitoroitavaksi. Haittapuolena tilattomuus-rajoitteen noudattamisessa on lisääntyvä verkon kuormitus. Tämän lisäksi istunnon tilasta huolehtimisen siirtäminen asiakassovellukselle vähentää palvelimen kykyä kontrolloida järjestelmän toiminta yhtenäiseksi erilaisten asiakassovellusten ollessa käytössä, koska järjestelmän yhtenäinen toiminta tulee riippuvaiseksi asiakassovelluksen oikeanlaisesta toteutuksesta.

2.1.3. Välimuistin käyttö -rajoite

Kolmas pakollinen rajoite on välimuistin käyttö [5 s. 79-81]. Välimuistin käyttö -rajoitteessa palvelimen lähettämiin vastauksiin asiakkaalle on sisällyttävä tieto, että voiko niitä säilöä välimuistiin. Noudattamalla välimuistin käyttö-rajoitetta voidaan verkon kuormitusta vähentää ja kyselyiden keskimääräistä vasteaikaa pienentää. Tämän rajoitteen noudattamisessa on kuitenkin riskinä datan luotettavuuden heikkeneminen, koska välimuistissa oleva data voi vanhentua.

2.1.4. Yhdenmukainen rajapinta -rajoite

Neljäs pakollinen rajoite on yhdenmukainen rajapinta [5 s. 81-82]. Yhdenmukainen rajapinta -rajoite vaatii järjestelmän eri komponenteilla olevan yhdenmukaiset rajapinnat, mikä vähentää järjestelmän arkkitehtuurin monimutkaisuutta sekä lisää komponenttien vuorovaikutuksen läpinäkyvyyttä. Haittapuolena tämän rajoitteen noudattamisessa on datan liikkuminen komponenttien välillä aina samassa standardoidussa muodossa, mikä saattaa olla epäoptimaalinen muoto datan käyttötarkoitusta varten. Tähän rajoitteeseen sisältyy neljä erilaista tarkentavaa rajoitetta: resurssien tunnistaminen, resurssien manipulointi niiden esitysten kautta,

itseselitteiset viestit sekä HATEOAS-periaatteen (Hypermedia As The Engine Of Application State) hyödyntäminen.

REST-ohjelmistoarkkitehtuurissa käytetään informaatiosta abstraktiota nimeltä “resurssi” [5 s. 88-90]. Resurssilla voidaan tarkoittaa mitä tahansa informaatiota, joka voidaan erikseen nimetä. Resurssi voi olla esimerkiksi kuva, palvelu tai kokoelma muista resursseista. Resurssilla ei kuitenkaan tarkoiteta itsessään resurssin sisältöä, vaan resurssi on abstrakti osoitin johonkin entiteettiin, joka voi vaihtua ajan kuluessa. Jokaisella resurssilla on oltava yksiselitteinen tunniste, joka mahdollistaa sen osoittamisen ja käytön eri REST-järjestelmän komponenttien välillä.

REST-järjestelmässä jokaiselle resurssille on määritelty esitys [5 s. 90-92]. Resurssin esitys sisältää varsinaisen datan sekä metadatan. Metadata sisältää esityksen metadatan, kontrollidatan ja tarvittaessa resurssin metadatan, joka ei ole riippuvainen esityksestä. Metadata koostuu avain-arvo -pareista. Esityksen metadata voi esimerkiksi sisältää tiedon esityksen varsinaisen datan tyypistä ja kontrollidata voi esimerkiksi sisältää tiedon, että sallitaanko esityksen tallentaminen välimuistiin. Jos resurssilla on useita esityksiä, voidaan niistä sopiva valita mediatyypin neuvottelulla. Resurssin esityksen avulla voivat järjestelmän eri komponentit käyttää resurssia haluamallaan tavalla, kuten esimerkiksi suorittaa resurssin tilan hakemisen tai asettamisen.

REST-järjestelmässä viestien on oltava itseselitteisiä. Viestin täytyy sisältää kaikki tieto, joka tarvitaan viestin tulkitsemiseen. Tämä vaatii, että viestissä resurssin esityksen metadatassa mediatyyppi ja kontrollidata sisältävät tiedot, joiden perusteella viestin vastaanottajan on mahdollista prosessoida viesti [5 s. 91]. Itseselitteiset viestit -rajoite liittyy vahvasti tilattomuus-rajoitteeseen, joka myös vaatii jokaisen viestin olevan järjestelmän tilasta riippumaton itsenäinen kokonaisuutensa. Itseselitteiset viestit antavat myös mahdollisuuden erilaisille välikomponenteille prosessoida niiden läpi liikkuvia viestejä.

HATEOAS-periaate, eli hypermedian käyttö sovelluksen tilakoneena on REST-ohjelmistoarkkitehtuurityylin selkeästi muista ohjelmistoarkkitehtuurityyleistä erottava rajoite. Tämä rajoite vaatii, että palvelin tarjoaa asiakkaalle dynaamisesti tietoa mahdollisista toiminnoista resurssin esitykseen sisältyvän metadatan avulla, ja asiakasohjelma ohjaa toimintaansa saamiensa vaihtoehtojen perusteella [11]. Asiakassovellus ei siis tarvitse rajapinnan alkupisteen lisäksi mitään muuta ulkopuolista informaatiota rajapinnasta, vaan palvelimen vastaukset sisältävät kaiken tarvittavan tiedon rajapinnan resursseista ja niiden käytöstä. HATEOAS-periaate antaa asiakassovellukselle mahdollisuuden automaattisesti kartoittaa koko rajapinta ja kaikkien resurssien toiminnot.

2.1.5. Kerroksittainen järjestelmä -rajoite

Viides pakollinen rajoite on kerroksittainen järjestelmä [5 s. 82-84]. Kerroksittaisessa järjestelmässä jokainen järjestelmän komponentti kuuluu kerrokseen, joka on vuorovaikutuksessa ainoastaan välittömästi sen yläpuolella tai alapuolella olevien kerrosten kanssa. Tämä mahdollistaa esimerkiksi vanhentuneen palvelun kapseloinnin rakentamalla sen päälle uudistettu rajapinta. Kerroksittaisen järjestelmän rajoitetta noudattavaan järjestelmään voi esimerkiksi palvelimen ja asiakkaan välille asettaa palomuureja tai välityspalvelimia ilman, että palvelimen tai asiakkaan

rajapintoja täytyisi muokata sitä varten. Asiakas ei voi, eikä sen tarvitse tietää, onko yhteys muodostettu suoraan todelliselle palvelimelle, tai onko yhteyden välissä esimerkiksi käytössä kuormaa tasaava välityspalvelin. REST-ohjelmistoarkkitehtuurityyliä noudattavassa järjestelmässä välikerroksen komponentit voivat myös muokata viestien sisältöä, koska jokainen viesti noudattaa viestien itseselitteisyyden rajoitetta. Haittapuolena kerroksittaisen järjestelmän rajoitetta noudattaessa on kasvava tarve datan prosessoinnille kerroksien lisääntyessä, mikä johtaa keskimääräisten vasteaikojen pitenemiseen. Tätä haittaa on mahdollista kompensoida välimuistin käytöllä järjestelmän eri kerroksissa.

2.1.6. Ladattava koodi -rajoite

Kuudes rajoite on ladattava koodi [5 s. 84-85]. Tämä rajoite ei ole pakollinen rajoite toisin kuin viisi muuta rajoitetta. Ladattava koodi -rajoite antaa palvelimelle mahdollisuuden sisällyttää asiakkaalle lähetettävään vastaukseen koodia, jonka asiakas voi suorittaa. Tämä mahdollistaa asiakassovellusten yksinkertaisemman ja keveämmän toteutuksen, koska kaikkea sen toiminnallisuutta ei ole pakko toteuttaa. Ladattava koodi myös mahdollistaa toiminnallisuuden lisäämisen asiakasohjelmiin jälkikäteen, mikä helpottaa koko järjestelmän sisältämän toiminnallisuuden laajentamista. Ladattavan koodin hyödyntäminen kuitenkin heikentää järjestelmän toiminnan läpinäkyvyyttä, mikä on syy kyseisen rajoitteen ei-pakollisuuteen.

2.2. HTTP-protokolla

HTTP (Hypertext Transfer Protocol), eli hypertekstin siirtoprotokolla, on sovelluskerroksen tiedonsiirtoon tarkoitettu protokolla hypermediajärjestelmille. HTTP on yksi webin tärkeimpiä protokollia ja sen tyypillinen käyttötapaus on verkkosivun tai binääridatan, kuten kuvien, siirtäminen palvelimelta asiakasohjelmalle [10]. Fielding kehitti REST-ohjelmistoarkkitehtuurityylin HTTP:n luomisen inspiroimana kuvaamaan ideaalista tapaa, jolla webin pitäisi toimia [5 s. 148], joten ei ole yllättävää, että HTTP soveltuu käytettäväksi REST-ohjelmistoarkkitehtuurityylin mukaisissa järjestelmissä.

HTTP perustuu asiakas-palvelin -arkkitehtuurimalliin. Asiakas lähettää palvelimelle HTTP-pyynnön, johon palvelin vastaa HTTP-vastauksella. HTTP-pyyntö sisältää metodin, URI:n (Uniform Resource Identifier), HTTP-protokollan version, pyynnön otsikkotiedot ja viestin varsinaisen sisällön. HTTP-vastaus on samankaltainen kuin HTTP-pyyntö. HTTP-vastaus sisältää tilakoodin, HTTP-protokollan version, otsikkotiedot sekä viestin varsinaisen sisällön [12].

HTTP-pyyntö sisältää URI:n [13]. URI on määritelmänsä mukaan kompakti merkkijono, jonka avulla resurssi voidaan identifioida. URL (Uniform Resource Locator) on URI:n alatyyppi, joka sisältää tiedon resurssin sijainnista kuvailemalla, kuinka resurssi on mahdollista hakea. URL:n tapauksessa merkkijonoon sisältyy tieto resurssin verkkosijainnista ja käytettävästä protokollasta [13 s. 7].

HTTP-pyynnön standardoituja metodeja ovat GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE ja CONNECT [14 s. 21-33].

- GET-metodi on ensisijainen tapa noutaa haluttu resurssin esitys palvelimelta.
- HEAD-metodin on tarkoitus antaa täsmälleen samanlainen vastaus asiakkaalle kuin GET-metodin, mutta palvelimen lähettämä vastaus ei sisällä viestin varsinaista sisältöä. HEAD-metodia voidaan käyttää esimerkiksi resurssin olemassaolon varmistamiseen tai otsikkotietojen (kuten viimeisimmän muokkausajan) selvittämiseen ilman yhteyttä enemmän kuormittavaa resurssin esityksen siirtämistä palvelimelta asiakkaalle.
- POST-metodia käytettäessä resurssia pyydetään prosessoimaan pyyntöön sisällytetty esitys. Prosessointi tapahtuu resurssin itse päättämällä tavalla, joka voi esimerkiksi tarkoittaa uuden resurssin luomista.
- PUT-metodia käytetään resurssin tilan päivittämiseen. PUT-metodi on samankaltainen kuin POST-metodi. Keskeinen ero näiden kahden metodin välillä on seuraava: POST-metodia käytettäessä pyyntöön sisältyvä resurssin esitys prosessoidaan resurssin määrittelemällä tavalla, kun taas PUT-metodia käytetään korvaamaan olemassa olevan resurssin tila pyynnön sisältämällä resurssin esityksellä.
- DELETE-metodia käytetään resurssin poistamiseen rajapinnasta.
- CONNECT-metodilla voidaan pyytää välityspalvelimelta tunnelin muodostamista kohdepalvelimelle.
- OPTIONS-metodilla voidaan kysyä palvelimelta sallittuja metodeja tietylle resurssille.
- TRACE-metodia käytettäessä palvelin palauttaa saamansa viestin sellaisenaan takaisin asiakkaalle. Tätä metodia voidaan käyttää hyödyksi vianmääritystarkoituksissa.

PATCH-metodi lisättiin HTTP-protokollaan vuonna 2010 [15]. PATCH-metodia käytetään resurssin osittaiseen päivittämiseen. Toisin kuin PUT-metodi, PATCH-metodi ei vaadi kokonaista resurssin esitystä, jolla resurssin tila korvataan. PATCH-metodia käytettäessä pyyntö sisältää ohjeet, kuinka palvelimen pitäisi muokata resurssin tilaa, jotta se vastaisi uutta haluttua tilaa.

GET-, HEAD- ja OPTIONS-metodit luokitellaan turvallisiksi metodeiksi, koska niiden käyttäminen ei tuota resurssin tilassa muutoksia. Näitä metodeja sekä metodeja PUT ja DELETE kutsutaan idempotenteiksi, koska niiden suorittaminen useammin kuin kerran ei muuta saatua lopputulosta.

HTTP-protokollassa vastausten tilakoodit (käytetään myös termiä “vastauskoodit”) ilmaistaan kolminumeroisella luvulla. Tilakoodit jakautuvat tämän luvun ensimmäisen numeron mukaan viiteen erilaiseen ryhmään. Tilakoodin kaksi viimeistä numeroa tarkentavat tilakoodin merkitystä tilakoodiryhmän sisällä. Vastausten tilakoodien on tarkoitus saada pyynnön lähettänyt asiakas ymmärtämään mitä on tapahtunut palvelimen suoritettaessa HTTP-pyyntöä. HTTP-vastauksen tilakoodeja voi myös laajentaa, eli palvelimen on mahdollista käyttää muitakin tilakoodeja kuin valmiiksi määriteltyjä. Asiakkaan ei ole pakko osata tulkita jokaista tilakoodia, ja asiakas voi

tulkita itselleen tuntemattoman tilakoodin samaksi kuin kyseisen tilakoodiryhmän ensimmäisen tilakoodin (esimerkiksi tilakoodi 418 voidaan tulkita tilakoodina 400). Vastaus sisältää yleensä viestirungossaan tilakoodin merkitystä selittävän esityksen [14 s. 47-64].

- Ryhmän 1xx tilakoodit ovat tarkoitettu informatiivisiksi. Informatiiviset tilakoodit ilmaisevat yhteyden tai pyynnön senhetkistä tilaa, joka lähetetään vastauksessa asiakkaalle ennen kuin koko pyyntö on prosessoitu.
- Ryhmän 2xx tilakoodit kuvaavat onnistunutta pyynnön toteutusta.
- Ryhmän 3xx tilakoodit ilmaisevat tarvetta uudelleenohjaukselle asiakkaan toimesta, jotta alkuperäinen pyyntö voidaan toteuttaa.
- Ryhmän 4xx tilakoodit on tarkoitettu kuvaamaan tilannetta, jossa asiakkaan lähettämä pyyntö on virheellinen.
- Ryhmän 5xx tilakoodit kuvaavat tilannetta, jossa palvelin ei voi toteuttaa asiakkaan pyyntöä sisäisen virheen vuoksi.

HTTP-protokollassa on määritelty erilaisia otsikkotietoja sekä HTTP-pyynnöille että HTTP-vastauksille. Näiden otsikkotietojen avulla pyyntöön tai vastaukseen voidaan sisällyttää lisäinformaatiota, joka ei käy ilmi muusta viestistä. HTTP-pyynnössä otsikkotiedot voivat esimerkiksi sisältää autentikaatiotietoja tai kertoa asiakkaan mieluisimman mediatyypin vastaukselle. HTTP-vastauksissa otsikkotiedot voivat esimerkiksi antaa lisätietoja palvelimesta tai resurssien sijainnista. Otsikkotiedot ovat rakenteeltaan joukko avain-arvopareja. Esimerkkinä hyödyllisistä otsikkotiedoista ovat esimerkiksi “Accept”- ja “Content-Type”-otsikkotiedot, joita käytetään hyödyksi valitessa mediatyypiltään sopivaa resurssin esitystä. Viestit voivat myös sisältää itse määriteltyjä otsikkotietoja. IANA (Internet Assigned Numbers Authority)[1] vastaa maailmanlaajuisesti HTTP-otsikkotietojen rekisterin ylläpidosta [14 s. 33-47,64-73].

2.3. REST-pohjaiset web-rajapinnat

REST-ohjelmistoarkkitehtuurityyli on saavuttanut suosiota web-rajapintojen kehityksessä. “ProgrammableWeb”-sivustolle rekisteröidyistä web-rajapinnoista yli neljä viidesosaa on luokiteltu REST-ohjelmistoarkkitehtuurityyliseksi vuonna 2017 [3]. Myös Cloud Elements kertoo raportissaan REST-ohjelmistoarkkitehtuurityylin (mutta ilman HATEOAS-periaatteen noudattamista) olevan suosituin tyyli yritysten tarjoamissa tai käyttämissä rajapinnoissa [4].

REST-pohjaisten rajapintojen toteutuksen on havaittu jättävän toteuttamatta REST-ohjelmistoarkkitehtuurityylin rajoitteita, mikä on nostattanut kritiikkiä aiheesta. On havaittu, että monet REST-pohjaiseksi itseään väittävät palvelut eivät todellisuudessa noudata REST-ohjelmistoarkkitehtuurityylin rajoitteita [16]. Myös REST-ohjelmistoarkkitehtuurityylin keksijä Roy Fielding on kritisoinut ohjelmistokehittäjiä, jotka kutsuvat rakentamiaan web-rajapintoja “REST-rajapinnoiksi”, vaikka ne

[1] https://www.iana.org/

eivät toteuta REST-ohjelmistoarkkitehtuurityyliin kuuluvaa pakollista HATEOAS-periaatetta. [11] [17].

Leonard Richardson on kehittänyt nelitasoisen mallin (Richardson Maturity Model) kuvaamaan rajapintojen "kypsyyttä" REST-ohjelmistoarkkitehtuurityylin toteutuksessa [18]. Jokainen taso edellyttää myös alempien tasojen vaatimuksien toteuttamista, ja tason 3 toteuttaminen on edellytys REST-ohjelmistoarkkitehtuurityylin rajoitteiden noudattamiselle.

- Taso 0 tarkoittaa, että rajapinta hyödyntää HTTP-protokollaa ainoastaan väylänä viestien lähettämiseen ja vastaanottamiseen, eikä rajapintaa ole jaettu itsenäisiin resursseihin.
- Tasolla 1 on rajapinta jaettu itsenäisiin resursseihin.
- Tasolla 2 resurssien käsittelyyn käytetään tapahtuvaa toimintaa yhtenevästi kuvaavia metodeja. Esimerkiksi HTTP:n GET-metodia käytetään noutamaan resurssin tila tai PUT-metodia käytetään muokkaamaan resurssin tilaa.
- Taso 3 toteuttaa HATEOAS-periaatteen lisäämällä rajapinnan lähettämiin vastauksiin dynaamisesti tietoa asiakassovellukselle mahdollisista toiminnoista.

2.4. REST-pohjaisten rajapintojen kuvauskielet

REST-rajapintojen kuvauskielien tarkoitus on toimia työkaluna, jolla voidaan tarjota jäsentynyt kuvaus REST-rajapinnasta. Tämän kuvauksen on tarkoitus olla sekä koneellisesti että ihmiselle ymmärrettävällä tavalla luettavissa sekä muokattavissa.

Kuvauskielen mukaisesta rajapinnan kuvauksesta on mahdollista tuottaa automaattisesti esimerkiksi rajapinnan toteuttavan palvelimen tai rajapintaa käyttävän asiakasohjelman ohjelmistokoodin runko. Ihmisluettavan dokumentaation automaattinen ja muodoltaan vakioitu generointi on mahdollista käyttämällä rajapinnan kuvausta. Kuvauskielestä riippuen voidaan generoida automaattisesti rajapintaan kohdistuvia testejä tai suorittaa automaattisesti REST-rajapinnan kuvauksessa esitettävien elementtien validointia [19] [8].

Rajapintojen kuvauskielien avulla tehty rajapinnan kuvaus tukee hyvin "API Desing first" -suunnittelumenetelmää (käytetään myös nimityksiä "schema-first" tai "API-driven-development") [7] käyttävää kehitysprosessia. Tässä prosessissa järjestelmän rajapinnat suunnitellaan ensimmäisenä ja niiden kuvaukset määritellään ja lukitaan ennen varsinaisen rajapinnan toimintalogiikan tai rajapintaa hyödyntävien komponenttien toteuttamista. Tällä tavalla menetellessä voidaan saavuttaa seuraavia etuja: Rajapinnan toteuttavia ja sitä hyödyntäviä komponentteja voidaan helposti kehittää samanaikaisesti, koska rajapinnan kaikki ominaisuudet ovat jo määriteltyjä. Aikaa ja rahaa on mahdollista säästää kehityksessä generoimalla automaattisesti ohjelmistokoodia rajapinnan kuvauksesta. Virheiden mahdollisuus kehityksen aikana vähenee ja kaikkien osapuolien tarpeet tulevat huomioiduksi, kun rajapinnan kuvaus suunnitellaan huolellisesti jokaista sidosryhmää kuunnellen [6] [8].

REST-rajapintojen kuvaukseen käytettäviä kuvauskieliä ovat esimerkiksi WSDL[2], OpenAPI-spesifikaatio [20], RAML[3] ja API Blueprint[4]. Näistä suosituimmaksi on noussut OpenAPI-spesifikaatio [4] [9].

2.5. Ohjelmistotestaus

Ohjelmistotestaus on tärkeä osa-alue ohjelmistokehityksessä. Ohjelmiston testaamisen päätarkoitus on löytää ohjelmistosta virheitä, jotta ne voitaisiin korjata ja ohjelmiston kokonaislaatu kohoaisi [21 s. 11]. Vaikka testauksen tavoitteena on löytää ja poistaa ohjelmistoista virheet, voidaan testauksen avulla todistaa ainoastaan virheiden löytyminen, ei niiden puutetta. Täysin kattavan testauksen saavuttaminen on todettu yleisessä tapauksessa mahdottomaksi, koska se vaatisi kaikkien teoreettisesti mahdollisten ohjelmiston syötteiden, tilojen ja suorituspolkujen testaamisen, mikä ei ole käytännössä saavutettavissa [22 s. 33-34]. Ohjelmistovirheillä voi olla raskaita taloudellisia seurauksia [23] ja kriittisillä toimialoilla ohjelmistovirheiden hintana saattavat olla ihmishenget. Näiden seikkojen johdosta teollisuudella ja akateemisella maailmalla on syytä olettaa olevan olemassa vahva kannustin käyttää aikaa ja rahaa ohjelmistojen testaamiseen, ohjelmistotestauksen tutkimukseen ja uusien työkalujen ja menetelmien kehittämiseen virheiden löytämiseksi ohjelmistoista.

Ohjelmistotestausta voidaan luokitella erilaisilla perusteilla. Yksi jakolinja voidaan tehdä staattisen ja dynaamisen testauksen välille:

- Staattisessa testauksessa testauksen kohdetta ei suoriteta, vaan sitä tutkitaan esimerkiksi katselmointien ja analyysityökalujen avulla. Tavoitteena staattisessa testauksessa on löytää esimerkiksi virheet tai muut ongelmakohdat ohjelmistokoodissa, tehdyissä suunnitelmissa tai vaatimusmäärittelyissä mahdollisimman aikaisessa vaiheessa kehitysprojektia, jolloin niistä seuraavia tulevaisuuden haittoja voidaan ehkäistä ennalta [22 s. 79].

- Dynaamisessa testauksessa suoritetaan ohjelmistokoodia käyttämällä erilaisia arvoja syötteenä ja varmistamalla, että saadut lopputulokset vastaavat kohteelle määriteltyjä vaatimuksia. Toisin kuin staattisessa testauksessa, jossa pyritään ehkäisemään tulevia ongelmia, dynaamisessa testauksessa etsitään virheitä, jotka ovat jo olemassa tutkittavassa ohjelmistokoodissa [22 s. 105-106].

Toinen perinteinen tapa luokitella ohjelmistotestausta on käyttää laatikkomallia. Laatikkomallissa voidaan ohjelmistotestaus jakaa kahteen eri luokkaan: mustalaatikkotestaus (“black-box-testaus”) ja lasilaatikkotestaus (“white-box-testaus”).

- Mustalaatikkotestauksessa testattava ohjelma nähdään sananmukaisesti mustana laatikkona: ohjelman sisäinen toiminta ja rakenne eivät ole tiedossa tai niitä ei oteta huomioon millään tavalla. Mustalaatikkotestauksessa pyritään löytämään ne olosuhteet, joissa ohjelma ei suoriudu vaaditulla tavalla. Koska tietoa

[2] https://www.w3.org/TR/wsdl20/
[3] https://raml.org/
[4] https://apiblueprint.org/

ohjelman sisäisestä toiminnasta ei hyödynnetä millään tavalla, käytetään testitapausten luonnissa hyväksi ainoastaan ohjelmalle asetettuja vaatimuksia, eli määritelmää, kuinka ohjelman pitäisi käyttäytyä. Mustalaatikkotestauksessa käytettäviä testaustekniikoita kutsutaan myös spesifikaatiopohjaisiksi, koska niissä luodaan testitapaukset esimerkiksi testikohteen alkuperäisistä vaatimusmäärittelyistä tai käyttötapauksien kuvauksista [21 s. 13] [22 s. 110].

- Lasilaatikkotestauksessa käytetään hyödyksi tietoa ohjelman sisäisestä toiminnan logiikasta, eli ohjelman lähdekoodin oletetaan olevan saatavilla. Lasilaatikkotestauksessa voidaan käyttää erilaisia testaustekniikoita, jotka pyrkivät luomaan testitapauksia siten, että testauksen kohde saavuttaa esimerkiksi mahdollisimman korkean lause-, haara-, ehto- tai polkukattavuuden [21 s. 14] [22 s. 145-146].

Ohjelmistotestauksen voidaan myös luokitella olevan funktionaalista tai ei-funktionaalista.

- Funktionaaliseen testaukseen kuuluu sellainen testaus, jossa tarkastellaan, toimiiko kohde halutulla tavalla, eli tuottaako kohde halutun lopputuloksen erilaisia syötteitä käyttämällä [22 s. 70].

- Ei-funktionaalisessa testauksessa tarkastelu kohdistuu toiminnallisuuden sijaan kohteen ominaisuuksiin. Kohdetta tarkastellaan pelkän syötteen ja tuloksen sijaan esimerkiksi sen tehokkuuden, luotettavuuden tai käytettävyyden näkökannasta [22 s. 72].

Ohjelmistotestausta voidaan ajatella luokiteltavaksi erilaisiin tasoihin sen mukaan, kuinka suurta kokonaisuutta testauksen kohteena käytetään.

- Yksikkötestaus (käytetään myös termejä “moduulitestaus” tai “komponenttitestaus”) tarkoittaa ohjelmiston pienempien osien, kuten yksittäisten funktioiden, testaamista löytämiseksi niistä niille asetettujen vaatimuksien vastaista toimintaa. Yksikkötestauksessa voidaan löytää ja osoittaa tehokkaasti virheiden tapahtumispaikat, koska vain pientä osaa ohjelmasta testataan kerrallaan [21 s. 70] [22 s. 42-43].

- Integraatiotestauksessa testataan laajempia kokonaisuuksia, joissa on liitetty yhteen useita jo yksikkötestauksen läpikäyneitä komponentteja. Integraatiotestauksessa tavoitteena on löytää virheitä komponenttien rajapinnoista ja komponenttien välisestä vuorovaikutuksesta [22 s. 50].

- Järjestelmätestauksen tehtävä on testata järjestelmän toimintaa täytenä kokonaisuutena. Järjestelmätestaus poikkeaa yksikkö- ja integraatiotestauksesta siten, että se suoritetaan lopullisen järjestelmän käyttäjien ja järjestelmän toiminnan alkuperäisten tavoitteiden näkökulmasta, eikä teknisien vaatimusten näkökulmasta [21 s. 96] [22 s. 58].

Kaikki testikohteet eivät ole yhtä helposti testattavissa. Ohjelmistotestauksessa kohteen testattavuudella tarkoitetaan, kuinka tehokkaasti ja nopeasti haluttuja

ominaisuuksia tai funktionaalisuutta voidaan testata. Testattavuudella tarkoitetaan myös, onko kohteen toiminnallisuuksia tai ominaisuuksia mahdollista testata ensinkään. Ohjelmisto voi olla rakennettu niin, että tiettyjä rajapintoja ei voida suoraan käyttää tai soveltuvan testausympäristön rakentaminen voi olla liian hankalaa [22 s. 274] [24].

Ohjelmistotestauksessa tärkeä käsite on testitapaus. Testitapauksen ISO/IEC/IEEE 24765:2017-standardin mukainen määritelmä on “dokumentoitu määritelmä syötteistä, odotetuista tuloksista ja testiprosessista yksittäiselle testille”. Testitapaus on yksittäinen ydinyksikkö, joiden kokonaisuudesta testauksen katsotaan muodostuvan [24]. Monen testitapauksen ryhmä muodostaa testijoukon (käytetään myös termiä “testiajo”), jossa on mahdollista, että yhden testin lopputulos voi esimerkiksi muodostaa seuraavan testin syötteen. Yksi esimerkki tästä tilanteesta on merkinnän lisääminen tietokantaan ensimmäisessä testissä ja sen muokkaaminen halutulla tavalla toisessa testissä [22 s. 9-10].

Manuaalisessa testauksessa (josta käytetään myös termiä “manuaalitestaus”) suorittaa testaaja testitapauksia esimerkiksi laaditun käsikirjoituksen mukaisesti testattavalle kohteelle tarkoituksenaan löytää virheitä sen toiminnasta. Automatisoidussa testauksessa testaus voidaan suorittaa ilman vuorovaikutusta testaajan toimesta. Testaus ja sen suorittava automaatio on kuitenkin luotava ennen automatisoidun testauksen hyödyntämistä, mikä vaatii alkuinvestoinnin ennen testauksen aloittamista. Positiivisena puolena automaattisessa testauksessa nähdään asiantuntijoiden toimesta testien tehokas uudelleenkäyttäminen, toistettavuus, korkeampi saavutettu testauksen kattavuus ja nopeampi ja kustannustehokkaampi testien suorittaminen [25] [26].

Jatkuva integraatio (CI, Continuous Integration) on ohjelmistokehitysmenetelmä, jossa jokainen ohjelmiston kehittäjistä integroi tekemänsä työn osaksi suurempaa kokonaisuutta tiheällä aikavälillä. Jatkuvan integraation kehitysmenetelmässä käytetään apuna jatkuvan integraation työkaluja, jotka jokaisen lähdekoodin muutoksen jälkeen kääntävät lähdekoodin ja suorittavat ohjelmiston testauksen automaattisesti [27]. Jatkuvan integraation ohjelmistokehitysmenetelmä ja järjestelmät ovat saavuttaneet suosiota kehittäjien keskuudessa ja niiden on esimerkiksi havaittu esimerkiksi ehkäisevän virheitä ohjelmistojen koontiversioissa ja tuomaan esille ohjelmistojen virheet nopeammin [28]. Esimerkkejä jatkuvan integraation työkaluista ovat Jenkins[5] ja GitLab CI[6].

2.6. Web-rajapintojen testaustyökalut

Web-rajapintojen testaamiseen on olemassa erilaisia testaustyökaluja, jotka ominaisuuksistaan riippuen esimerkiksi helpottavat testitapausten luontia, testien suorittamista, testauksen automatisointia tai rajapinnan toiminnan monitorointia. Enemmistö työkaluista perustuu mustalaatikkotestaukseen, koska pääpaino rajapinnan testauksessa on toiminnallisuuden varmistamisessa ja virheiden etsimisessä, eikä rajapinnan taustalla toimivista sovelluksista välttämättä ole saatavilla tarkkaa tietoa. Pääosin työkalut keskittyvät funktionaaliseen testaukseen, mutta osa

[5] https://jenkins.io/

[6] https://about.gitlab.com/stages-devops-lifecycle/continuous-integration/

työkaluista tarjoaa mahdollisuuksia myös ei-funktionaaliseen testaukseen, kuten kuormitustestaukseen.

Web-rajapintojen testauksen on havaittu olevan tärkeä prioriteetti kehittäjien näkökulmasta [9]. Rajapintojen kehittäjille ilmeinen syy testaustyökalujen käytölle on tehdä testauksesta varmempaa, nopeampaa, kustannustehokkaampaa ja mahdollistaa testauksen suorittaminen ilman laajaa teknistä tuntemusta, eli toisin sanoen säästää aikaa, vaivaa ja rahaa.

Useat työkalut pyrkivät tarjoamaan rajapinnan testausta suorittavalle testaajalle yksinkertaistetun tavan luoda, tallentaa ja suorittaa testitapauksia. Työkalut voivat esimerkiksi tarjota valmiin ohjelmistokirjaston tai käyttöliittymän HTTP-kutsujen luomiselle ja saatujen vastauksen sisällön tarkastelulle. Tällaisia ominaisuuksia sisältäviä työkaluja ovat esimerkiksi Postman[7], SoapUI[8] (tunnetaan myös nimellä ReadyAPI), APIFortress[9], hippie-swagger[10], pyresttest[11], Airborne[12], HttpMaster[13], REST-assured[14], Tricentis[15], Katalon[16], Dredd[17], vREST[18], TnT-Fuzzer [29], APIFuzzer [30], Meqa [31], Schemathesis [32], “Got Swag?” [33] ja Insomnia[19]. Useimmat mainitut työkalut tarjoavat mahdollisuuksia luoda testitapauksista helposti käsiteltäviä kokoelmia ja käyttää niitä tehokkaasti osana jatkuvan integraation järjestelmiä. Jotkin mainituista työkaluista ovat kykeneviä avustamaan testaajaa testauksen luonnissa generoimalla valmiita pohjia testitapauksille tai jopa valmiita testitapauksia esimerkiksi rajapinnan kuvaustiedostoista, jotka noudattavat tuettua rajapinnan kuvauskieltä, kuten OpenAPI-spesifikaatiota.

Muutamat työkalut tarjoavat rajapintojen testausta SaaS-palveluna (Software as a Service). Käyttämällä palvelua asiakas voi luoda testauksen omalle rajapinnalleen, minkä jälkeen palvelu testaa rajapintaa säännöllisesti virheiden varalta sekä esimerkiksi monitoroi rajapinnan suorituskykyä ja käytettävyysaikaa. Tämänkaltaisia työkaluja ovat esimerkiksi API Science[20], Ping-API[21], Assertible[22] ja Runscope[23].

Rajapintojen testaamiseen on työkaluja, jotka testaavat rajapinnan ei-funktionaalisia ominaisuuksia. Esimerkiksi työkalut JMeter[24] ja SoapUI sisältävät erilaisia ominaisuuksia kuormitus- ja suorituskykytestaukseen. Turvallisuustestaukseen rajapinnoille voidaan käyttää erilaisia turvallisuusskannereiksi kutsuttuja työkaluja, jotka pystyvät etsimään haavoittuvuuksia tai riskejä rajapinnoista automaattisesti

[7] https://www.postman.com/
[8] https://smartbear.com/product/ready-api/overview/
[9] https://apifortress.com
[10] https://github.com/CacheControl/hippie-swagger
[11] https://github.com/svanoort/pyresttest
[12] https://github.com/brooklynDev/airborne
[13] https://www.httpmaster.net
[14] http://rest-assured.io/
[15] https://www.tricentis.com
[16] https://www.katalon.com
[17] https://dredd.org
[18] https://vrest.io
[19] https://insomnia.rest/
[20] https://www.apiscience.com/
[21] https://ping-api.com
[22] https://assertible.com/
[23] https://www.runscope.com
[24] https://jmeter.apache.org/

tai käyttäjän avustamana. Tällaisia työkaluja ovat esimerkiksi SoapUI:n sisältämät haavoittuvuuksien etsintätyökalut, Astra[25], OWASP Zed Attack Proxy[26] tai Automatic API Attack Tool[27].

Edellä viitataan joukkoon erilaisia web-rajapintojen testaamiseen kehitettyjä työkaluja, mutta annetut esimerkit työkaluista ovat vain pieni joukko kaikista tarkoitukseen kehitetyistä työkaluista. Web-rajapinnoille tuotetun testauksen evaluoimiseen ei ole saatavilla samalla tavalla runsasta joukkoa työkaluja, joten tältä osin mahdollisuuksia kehitykselle voidaan sanoa olevan olemassa.

[25]https://github.com/flipkart-incubator/Astra
[26]https://github.com/zaproxy/zaproxy/
[27]https://github.com/imperva/automatic-api-attack-tool

3. OPENAPI-SPESIFIKAATION MUKAISEN RAJAPINNAN TESTAUKSEN KATTAVUUDEN MÄÄRITTÄMINEN

Tässä luvussa esitellään OpenAPI-spesifikaatio (OAS) sekä työssä käytettävä tapa mitata rajapinnan testauksen kattavuutta suhteessa OAS:n mukaiseen kuvaustiedostoon.

OAS [20] on määritelmä REST-rajapinnan kuvaukseen tarkoitetulle kuvauskielelle, jolla voidaan kuvata rajapinnan käyttömahdollisuudet sekä ihmis- että koneluettavassa muodossa.

OAS:n mukaisen rajapinnan testauksen kattavuuden mittaamiseen voidaan hyödyntää rajapinnan kuvaustiedostossa määriteltyjen elementtien käytetyksi tulemista. Tarkastelemalla testauksen aikana tapahtuneita rajapinnan päätepisteiden, operaatioiden, vastausten ja parametrien käyttöä voidaan havaita testauksen puutteita tai ongelmakohtia.

3.1. OpenAPI-spesifikaatio

OpenAPI-spesifikaatio [20] (josta käytetään myös termiä OpenAPI/Swagger tai lyhennettä OAS) on määritelmä REST-rajapinnan kuvaukseen tarkoitetulle kuvauskielelle. Kuvauskielen tarkoituksena on tarjota kieliriippumaton kuvaus REST-rajapinnasta, joka on sekä koneellisesti että ihmisten luettavissa. OAS:n mukaisen rajapinnan kuvaustiedosto tarjoaa REST-rajapinnan käyttäjälle mahdollisuuden nähdä rajapinnan kaikki käyttömahdollisuudet ilman tarvetta tutustua esimerkiksi palvelun ohjelmistokoodiin sen toiminnan selvittämiseksi. Kuvaustiedostoa voidaan käyttää lähteenä erilaisille työkaluille, jotka tuottavat automaattisesti ohjelmistokoodia, rajapintaan kohdistuvia testejä tai dokumentaatiota. OAS on noussut suosituksi ohjelmistokehittäjien keskuudessa [4] [9].

3.1.1. OpenAPI-spesifikaation historia

OAS:n historia ulottuu vuoteen 2010, jolloin yritys nimeltä Wordnik aloitti Swagger-spesifikaation kehityksen. Swagger-spesifikaatio julkaistiin avoimen lähdekoodin lisenssillä vuotta myöhemmin [34]. Vuonna 2015 SmartBear Software otti haltuunsa Swagger-spesifikaation yritykseltä nimeltä Reverb Technologies [35]. Samana vuonna SmartBear Software ilmoitti perustavansa yhdessä usean muun tahon kanssa (3Scale, Apigee, Capital One, Google, IBM, Intuit, Microsoft, PayPal ja Restlet) yhteenliittymän nimeltä OpenAPI Initiative (OAI), joka asetettiin toimimaan Linux Foundation -säätiön alaisuuteen. Perustamisen yhteydessä SmartBear Software lahjoitti Swagger-spesifikaation Linux Foundation -säätiölle. Swagger 2.0-spesifikaatio ja OpenAPI 2.0-spesifikaatio (käytetään myös lyhennettä OAS 2.0) ovat täysin samanlaisia, ja niihin viitatessa saatetaan käyttää termiä OpenAPI/Swagger. Kuvauskielestä julkaistiin versio 3.0 vuonna 2017 [36] ja päivitys versioon 3.0.2 julkaistiin vuonna 2018 [20].

3.1.2. OpenAPI-spesifikaation rakenne

OAS:n avulla on mahdollista kuvata REST-ohjelmistoarkkitehtuurityylin mukaisia rajapintoja [20] [37]. OAS ei kuitenkaan sisällä mitään sääntöjä rajapinnalle, jotka pakottaisivat rajapinnan noudattamaan REST-ohjelmistoarkkitehtuurityylin rajoitteita. OAS tukee ainoastaan HTTP-protokollan avulla käytettävien rajapintojen kuvausta. Rajapinnan suunnittelijat ja kehittäjät voivat siis käyttää kuvauskieltä HTTP-protokollaa hyödyntävien rajapintojen kuvaamiseen oman harkintansa mukaan, olipa kyseessä REST-ohjelmistoarkkitehtuurityyliä noudattava rajapinta tai ei.

OAS mahdollistaa rajapinnan kuvaamisen YAML (YAML Ain't Markup Language) - tai JSON (JavaScript Object Notation) -tiedostomuotoisella kuvaustiedostolla. OAS:n mukainen kuvaustiedosto sisältää esimerkiksi rajapinnan metatiedot (kuten otsikon, tekijän yhteystiedot tai palvelimen verkkosijainnin), tiedot rajapinnan rakenteesta sekä tiedot, millaisilla HTTP-kutsuilla rajapintaa voi kutsua. Samoin voidaan kyseisten kutsujen odotetuista vastauksista määritellä esimerkiksi HTTP-vastauskoodit ja HTTP-vastausten sisällön muoto.

Kuva 1 esittää osan usein OAS:n esimerkkinä hyödynnetyn "Swagger Petstore"-rajapinnan YAML-muotoisesta kuvaustiedostosta. Kuvasta käy ilmi OAS:n mukainen esitystapa rajapinnan päätepisteiden kuvaamiselle: Jokainen rajapinnan päätepiste (eli resurssi) sisältää operaatiot (eli resurssille käytettävissä olevat HTTP-metodit), joilla kyseistä päätepistettä voidaan kutsua. Jokainen operaatio sisältää sen mahdollisten parametrien määrittelyt sekä kutsun mahdollisten vastauksien määritelmät.

OAS:ssa määritellyt parametrit sisältävät esimerkiksi tietoja niiden nimestä, muodosta, käytön pakollisuudesta ja sijainnista HTTP-kutsussa. Yksittäiselle päätepisteelle voidaan määritellä myös yleisiä parametreja, jolloin kyseisen päätepisteen kaikkien operaatioiden katsotaan sisältävän kyseinen parametri. Parametrin sijainnin tyypille HTTP-kutsussa on OAS 2.0:ssa viisi erilaista vaihtoehtoa [37]:

- Tyyppi "path" tarkoittaa parametrin olevan osa HTTP-kutsun URL-polkua. Esimerkiksi URL "http://esimerkki.com/tilaukset/12345" voi sisältää parametrin arvolla "12345".

- Tyyppi "query" merkitsee parametrin sisältymistä HTTP-kutsun URL-polun jälkeiseen "query"-osuuteen avain-arvo-parina. Esimerkiksi URL "http://esimerkki.com/tilaukset?tyyppi=valmis" sisältää parametrin nimeltä "tyyppi" arvolla "valmis".

- Tyyppi "header" tarkoittaa parametrin sisältyvän HTTP-kutsun otsikkotietoihin avain-arvo-parina.

- Tyyppi "formData" merkitsee parametrien sisältyvän HTTP-kutsun viestirunkoon lomakedatan muodossa. Tässä tilanteessa HTTP-kutsun otsikkotiedoissa käytetään esimerkiksi sisällön mediatyyppiä "application/x-www-form-urlencoded" kertomaan datan tarkempi muoto HTTP-viestirungossa.

- Tyyppi "body" kertoo parametrin sijaitsevan HTTP-kutsun viestirungossa. Tyypin "body" ja "formData" käyttö saman operaation parametreissa ovat

toisensa poissulkevia, koska molemmat varaavat HTTP-kutsun viestirungon omaan käyttöönsä.

```yaml
  externalDocs:
    description: "Find out more about our store"
    url: "http://swagger.io"
schemes:
- "https"
- "http"
paths:
  /pet:
    post:
      tags:
      - "pet"
      summary: "Add a new pet to the store"
      description: ""
      operationId: "addPet"
      consumes:
      - "application/json"
      - "application/xml"
      produces:
      - "application/xml"
      - "application/json"
      parameters:
      - in: "body"
        name: "body"
        description: "Pet object that needs to be added to the store"
        required: true
        schema:
          $ref: "#/definitions/Pet"
      responses:
        "405":
          description: "Invalid input"
      security:
      - petstore_auth:
        - "write:pets"
        - "read:pets"
    put:
      tags:
      - "pet"
      summary: "Update an existing pet"
      description: ""
      operationId: "updatePet"
      consumes:
      - "application/json"
      - "application/xml"
      produces:
      - "application/xml"
      - "application/json"
      parameters:
      - in: "body"
        name: "body"
        description: "Pet object that needs to be added to the store"
        required: true
        schema:
          $ref: "#/definitions/Pet"
      responses:
        "400":
          description: "Invalid ID supplied"
        "404":
          description: "Pet not found"
```

Kuva 1. OAS 2.0:n mukainen YAML-tiedostomuotoinen kuvaustiedosto

OAS:ssa operaatioille määritellyt vastaukset sisältävät tiedot esimerkiksi kyseisen vastauksen HTTP-vastauskoodista, vastauksen viestirungon sisällöstä ja vastauksen otsikkotiedoista. OAS ei pakota rajapinnan kuvausta määrittelemään jokaista mahdollista vastausta kutsulle, mutta on suositeltavaa määritellä ainakin yksi vastaus onnistuneelle kutsulle sekä vastaukset kaikille ennakoitavissa oleville virhetilanteille. Operaatiolle on myös mahdollista määritellä vakiovastaus, jonka tarkoitus on kattaa jokainen HTTP-vastauskoodi, jota ei määritellä erikseen operaation vastauksissa [20] [37].

3.2. OpenAPI-spesifikaation mukaisen rajapinnan testauksen kattavuuden määrittäminen

OAS:n mukaisten rajapintojen testauksen kattavuuden määrittelystä on tehty tutkimustyötä. Ed-douibi et al. mittaavat tutkimuksessaan [38] testauksen kattavuutta OAS:n mukaiselle rajapinnalle käyttämällä rajapinnan kuvauksen päätepisteitä, operaatioita, parametreja sekä tietotyyppien määritelmiä. Moniportaisen testauksen kattavuuden mallin REST-pohjaisille rajapinnoille esittävät Martin-Lopez et al. [39]. Esitetyssä mallissa huomioidaan rajapinnan päätepisteiden, operaatioiden, parametrien, mediatyyppien, erilaisten vastauskoodien, HTTP-vastauksien viestirunkojen ja operaatioiden erilaisten sekvenssien kattavuus. Vastaavaa kattavuuden mittausta toteuttavat työkalut vaikuttavat olevan harvinaisempia, koska ainoaksi esimerkiksi tämän työn kirjoitushetkellä löytyi työkalu SoapUI, joka sisälsi ominaisuuksia testauksen kattavuuden laskemisesta OAS:n mukaiselle kuvaustiedostolle[1].

Tässä työn osiossa esitellään tapa määritellä rajapinnan testauksen kattavuus OAS:n mukaista rajapinnan kuvaustiedostoa hyödyntämällä. Esiteltäviä mittareita tullaan hyödyntämään tämän työn myöhemmissä osioissa, joissa tullaan esittelemään testauksen kattavuutta mittaava työkalu sekä erilaisten testaustyökalujen tuottaman testauksen kattavuuden mittaukset. Näiden mittareiden tarkoituksena on kuvata, kuinka suurta osuutta rajapinnan päätepisteistä, metodeista, parametreista ja vastauksista on tullut katetuksi testauksen toimesta. Esiteltävät mittarit ovat hyvin samankaltaisia kuin Ed-douibi ja Martin-Lopez omissa tutkimuksissaan esittävät.

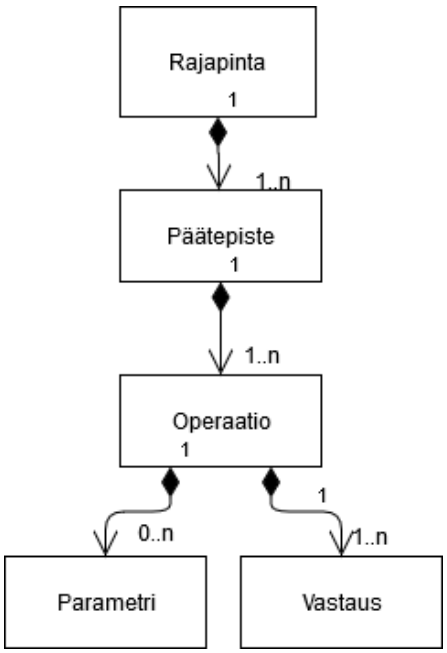
3.2.1. OpenAPI-spesifikaation mukaisen rajapinnan kattavuuden mittarit

Testauksen kattavuuden mittaamiseen käytetään tässä työssä seuraavaa periaatetta: lasketaan, kuinka monta rajapinnan päätepistettä, operaatiota, parametria tai vastausta on testauksen toimesta tullut käytetyksi suhteessa kaikkiin vastaaviin rajapinnan elementteihin. Testauksen kattavuutta voidaan tarkastella koko rajapinnan lisäksi yksittäisten päätepisteiden tai operaatioiden näkökulmasta.

OAS:n avulla rajapinta ja sen toiminta voidaan kuvata yksityiskohtaisesti. Määriteltäessä testauksen kattavuutta tarkastellaan rajapinnalle kuvaustiedostossa määriteltyjä päätepisteitä, operaatioita, vastauksia ja parametreja. Kuva 2 havainnollistaa yksinkertaistetusti näiden elementtien suhdetta: Rajapinta sisältää päätepisteitä. Päätepisteet sisältävät operaatioita. Operaatiot sisältävät vastauksia. Jokainen operaatio voi sisältää erilaisia parametreja.

OAS:ssa on mahdollista asettaa parametri päätepisteelle, jolloin se katsotaan sisältyväksi kaikkiin päätepisteen sisältämiin operaatioihin. Eri operaatiot voivat myös sisältää samoja yleisesti määriteltyjä parametreja. Kattavuuden määrittelemisen kannalta parametrit tulkitaan kuitenkin jokaisessa operaatiossa olevina erillisinä, toisistaan riippumattomana, parametreina. Kattavuuden tarkastelussa yksittäinen parametri siis muodostuu yksilöllisestä operaation ja parametrin yhdistelmästä, eikä pelkästään parametrin alkuperäistä määritelmää tarkastella sellaisenaan. Rajapinnan

[1] https://support.smartbear.com/readyapi/docs/testing/coverage/rest.html



Kuva 2. Pelkistetty kuvaus OAS:n mukaisen rajapinnan kuvaustiedoston elementtien suhteista

kuvauksessa on myös mahdollista käyttää samaa määriteltyä vastausta operaatioiden välillä, joten kattavuuden kannalta sitä käsitellään samalla tavoin kuin parametria, eli yksilöllisenä operaation ja vastauksen yhdistelmänä.

Rajapinnan testaukselle voidaan määrittää seuraavia kattavuuden mittareita:

- Rajapinnan päätepisteiden kattavuus. Kuvaa, kuinka monta päätepistettä rajapinnan kaikista päätepisteistä on käytetty vähintään kerran testauksen aikana.
- Rajapinnan operaatioiden kattavuus. Kuvaa, kuinka monta operaatiota rajapinnan kaikista operaatioista on käytetty vähintään kerran testauksen aikana.
- Rajapinnan vastausten kattavuus. Kuvaa, kuinka monta vastausta rajapinnan kaikista vastauksista on saatu vastauksena vähintään kerran testauksen aikana.
- Rajapinnan parametrien kattavuus. Kuvaa, kuinka monta parametria on käytetty kaikista rajapinnan parametreista.

Kattavuuden tarkastelua voidaan rajoittaa koskemaan rajapinnan yksittäistä päätepistettä koko rajapinnan sijaan. Yksittäisen päätepisteen testaukselle voidaan määrittää seuraavia kattavuuden mittareita:

- Päätepisteen operaatioiden kattavuus. Kuvaa, kuinka monta operaatiota kyseisen päätepisteen kaikista operaatioista on kutsuttu vähintään kerran testauksen aikana.
- Päätepisteen vastausten kattavuus. Kuvaa, kuinka monta vastausta kyseisen päätepisteen kaikista vastauksista on saatu vastauksena vähintään kerran testauksen aikana.
- Päätepisteen parametrien kattavuus. Kuvaa, kuinka monta parametria kyseisen päätepisteen kaikista parametreista on käytetty vähintään kerran testauksen aikana.

Kattavuuden tarkastelua voidaan rajoittaa koskemaan ainoastaan yksittäisen rajapinnan päätepisteen yksittäistä operaatiota. Yksittäisen operaation testaukselle voidaan määrittää seuraavia kattavuuden mittareita:

- Operaation vastausten kattavuus. Kuvaa, kuinka monta vastausta kyseisen operaation kaikista vastauksista on saatu vastauksena vähintään kerran testauksen aikana.
- Operaation parametrien kattavuus. Kuvaa, kuinka monta parametria kyseisen operaation kaikista parametreista on käytetty vähintään kerran testauksen aikana.

3.2.2. Kattavuuden mittaamisen edut ja ongelmakohdat

Tässä työn osiossa esitellyt kattavuuden mittarit sisältävät etuja ja ongelmakohtia.

Esitetyn tavan avulla voidaan havaita, milloin jokin tarkastelun kohteena olevasta elementeistä, eli päätepiste, operaatio, vastaus tai parametri, ei tule käytetyksi testauksen aikana. Kun testauksen on havaittu olevan rajapinnan päätepisteiden, operaatioiden, vastausten ja parametrien osalta katettu, voidaan olla varmoja, että mitään ilmeistä kohdetta ole jätetty täysin testaamatta. Kattavuutta kuvaamaan käytetty suhdeluku on helposti ymmärrettävissä oleva ja vertailukelpoinen mittari, jota voi käyttää kuvaamaan testauksen onnistumista esimerkiksi samaan rajapintaan kohdistuvien erilaisten testisarjojen välillä.

Vaikka elementtien kattavuuden tutkiminen voi osoittaa mahdolliset ilmeiset puutteet testauksessa, rajapinnan päätepisteen, operaation, vastauksen tai parametrin katettuna olo ei varsinaisesti kerro mitään muuta kuin sen olleen testauksen kohteena vähintään kerran. Yksikin kohteen käyttö riittää katetuksi merkitsemiseen, mutta tämänkaltaisen testauksen todellinen hyöty saattaa jäädä hyvin rajalliseksi, koska rajapinnan toteuttava ohjelmisto ei välttämättä tule perusteellisesti testatuksi. Ongelmaa voi pyrkiä rajoittamaan käyttämällä testauksessa esimerkiksi enemmän erilaisia kombinaatioita testattaville parametreille.

Kuvaustiedoston puutteellinen tarkkuus rajapinnan elementtien kuvaamisessa saattaa aiheuttaa ongelmia esitellyn mittaustavan käyttämiselle luotettavana mittarina. OAS:ssa kaikkia operaation mahdollisia vastauksia ei ole pakko määritellä erikseen, vaan on täysin kuvaustiedoston tekijän päätettävissä, kuinka tarkasti rajapinnan vastaukset määritellään. On myös mahdollista, että rajapinnan kuvaustiedostoa luodessa ei kaikkia mahdollisia vastauksia voi tietää etukäteen, joten ne on jätettävä määrittelemättä kokonaan. Operaation kuvaus voi esimerkiksi sisältää ainoastaan onnistuneen tilanteen vastauksen, esimerkiksi HTTP-vastauskoodin 200. Tässä tilanteessa kaikki virhetapaukset voivat jäädä määrittelemättä, vaikka ne olisivatkin mahdollisia tilanteita. Tämä voi johtaa tilanteeseen, jossa esitelty mittaustapa antaa täyden vastauksien kattavuuden operaatiolle, vaikka ainoastaan yksi yksinkertainen testi olisi suoritettu. OAS mahdollistaa myös vakiovastauksen määrittelyn operaatiolle. Jos vakiovastauskoodia käytetään esimerkiksi kattamaan kaikki mahdolliset tunnetut virhetapaukset, ei yksittäisten virhetapausten katetuksi tulemista ole mahdollista tarkastella. Näitä ongelmia on mahdollista rajoittaa etukäteen rajapinnan kuvauksen huolellisella suunnittelulla tai rajapinnan kuvauksen päivittämisellä tarpeen vaatiessa.

4. TYÖKALUN KUVAUS

Tässä luvussa esitellään työkalu “API Specification Coverage tool” (ASC) OpenAPI-spesifikaation mukaisten web-rajapintojen dynaamisen testauksen kattavuuden määrittelyyn rajapinnan kuvaustiedoston ja testien aikana tallennetun verkkoliikenteen avulla. ASC analysoi, mitä osuuksia rajapinnan kuvaustiedostosta on käytetty vertailemalla sitä tallennettuun verkkoliikenteeseen ja tuottaa analyysin tuloksena erilaisia raporttitiedostoja, jotka kuvaavat rajapinnan testauksen kattavuutta. ASC:a myös käyttää havaitsemaan ja raportoimaan tilanteet, joissa verkkoliikenteessä esiintyy rajapinnan kuvauksesta poikkeavia HTTP-kutsuja tai -vastauksia. Työkalua on mahdollista käyttää osana jatkuvan integraation testausjärjestelmää.

ASC on saatavilla GitHub-palvelussa[1].

4.1. Työkalun soveltuvuus erilaisiin käyttötarkoituksiin

ASC:lla on kolme erilaista mahdollista käyttötarkoitusta:

- Analysoida, mitkä elementit rajapinnan kuvauksessa ovat olleet testauksen kohteina, kuinka monta kertaa ja millä arvoilla. Tämän analyysin on tarkoitus kuvata rajapintaan kohdistuneen testauksen kattavuutta.
- Vertailla verkkoliikennettä rajapinnan kuvaukseen ja siten havaita mahdollisia ristiriitaisuuksia näiden kahden välillä. Havaitut ristiriitaisuudet voivat toimia tukena sekä rajapinnan toteutuksen että sen testitapausten kehittämisessä.
- ASC on mahdollista konfiguroida toimimaan osana jatkuvan integraation testausjärjestelmää. Työkalun ajonaikaista toimintaa määrittelevässä konfiguraatiotiedostossa voidaan määritellä ehtoja kahden edellä mainitun ominaisuuden suhteen. Mikäli konfiguraatiotiedostossa esitetyt ehdot rajapintaan kohdistuneiden testien kattavuudesta tai ristiriitaisuuksien esiintymisestä eivät täyty, työkalu lopettaa suorituksen käyttämällä yleistä virhettä merkitsevää poistumiskoodia. Tässä tilanteessa jatkuvan integraation testausjärjestelmän on mahdollista havaita, että suoritettu testaus ei ole täyttänyt sille määriteltyjä vaatimuksia.

ASC vaatii ainoastaan rajapinnan kuvaustiedoston ja testien aikana tallennetun verkkoliikenteen suorittaakseen analyysin. Tämä tarkoittaa, että työkalu on täysin riippumaton tavasta, jolla rajapinta, testisarjat rajapinnan testaamiseksi tai verkkoliikenteen tallentaminen on toteutettu. Työkalu tuottaa analyysin tuloksena sekä ihmis- että koneluettavia raportteja. Nämä työkalun ominaisuudet mahdollistavat työkalun käytön erilaisissa testausjärjestelmissä tai työkalun integroinnin muihin testauskäyttöön tarkoitettuihin työkaluihin.

4.2. Työkalun toiminta

ASC tuottaa analyysin rajapinnan testauksen kattavuudesta rajapinnan kuvaustiedoston ja testien aikana tallennetun verkkoliikenteen perusteella. Työkalu

[1] https://github.com/ouspg/ASC

lukee sisään rajapinnan kuvauksen ja tallennetun verkkoliikenteen. Tämän jälkeen työkalu laskee jokaiselle rajapinnan päätepisteelle, operaatiolle, parametrille ja vastaukselle käyttömäärät sekä etsii verkkoliikenteestä mahdollisia poikkeamia suhteessa rajapinnan kuvaustiedoston sisältämiin määrittelyihin. Lopuksi työkalu kirjoittaa lasketun analyysin raporttitiedostoihin ja lopettaa suorituksen tarvittaessa virheellisellä poistumiskoodilla, mikäli työkalun konfiguraatiotiedostossa määritellyt ehdot tälle täyttyvät.

4.2.1. Työkalun riippuvuudet

ASC suoritetaan Python 3.7 -komentotulkkia käyttämällä. Työkalu hyödyntää toiminnallisuutensa toteuttamiseen useita Python-ohjelmistokirjastoja suorituksensa aikana, kuten esimerkiksi prance, haralyzer, swagger-parser, openapi-spec-validator, regex, Jinja2 ja requests.

4.2.2. Työkaluun syötettävät parametrit

ASC edellyttää syötettäväksi kaksi pakollista tiedostoa työkalun käynnistyshetkellä: Rajapinnan kuvaustiedoston ja tallennetun verkkoliikenteen tiedoston. Valinnaisena syötettävänä parametrina voidaan työkalulle antaa erillinen konfiguraatiotiedosto, joka sisältää tarkempia työkalun toimintaa ohjaavia asetuksia.

Rajapinnan kuvaustiedosto on työkalun ensimmäinen pakollinen syötettävä parametri. Työkalu tukee OAS:n (2.0 ja 3.0) mukaisia rajapinnan kuvaustiedostoja. Kuvaustiedostot voivat olla sekä YAML- että JSON-tiedostomuodossa.

Tiedosto, joka sisältää tallennetun verkkoliikenteen on toinen työkalulle syötettävä pakollinen parametri. Työkalu tukee HTTP Archive Format -tiedostomuotoa (HAR).

4.2.3. Työkalun suorituksenaikaisen toiminnan kuvaus

Käynnistyessään ASC lukee sille parametreina syötetyt rajapinnan kuvaustiedoston ja tiedoston, joka sisältää tallennetun verkkoliikenteen. Tarvittaessa työkalu lukee myös annetun konfiguraatiotiedoston, mutta tämän puuttuessa käytetään asetuksina ennalta määritettyjä vakioarvoja. Rajapinnan kuvaustiedostosta poimitaan rajapinnan päätepisteet, operaatiot, parametrit ja vastaukset. Verkkoliikenteen sisältävä HAR-muotoinen tiedosto luetaan ja siitä poimitaan jokainen verkkoliikenteen HTTP-kutsu ja -vastaus, jotka sitten lajitellaan kuuluvaksi vastaavalle rajapinnan päätepisteelle ja operaatiolle.

Alkutietojen onnistuneen syöttämisen ja lajittelun jälkeen ASC aloittaa rajapinnan kuvauksen ja liikenteen vertailun. Jokaisen verkkoliikenteen kirjauksen, eli HTTP-kutsun ja -vastauksen, HTTP-vastauskoodi tutkitaan ja sitä vertaillaan rajapinnan kuvauksessa määriteltyihin vastauskoodeihin. Samoin analysoidaan jokainen HTTP-kutsu ja sen sisältöä vertaillaan rajapinnan kuvaustiedostossa määriteltyihin tietoihin. Tällä laskennalla saadaan selville jokaisen rajapinnan määritellyn päätepisteen, operaation, parametrin ja vastauksen käyttökertojen määrät sekä parametrien ja

vastauskoodien erilaiset arvot. Saaduista tuloksista luodaan analyysi koko rajapinnan testauksen kattavuudesta päätepisteiden, operaatioiden, vastausten ja parametrien osalta, sekä vastaava tarkastelu yksittäisten päätepisteiden sekä operaatioiden osalta.

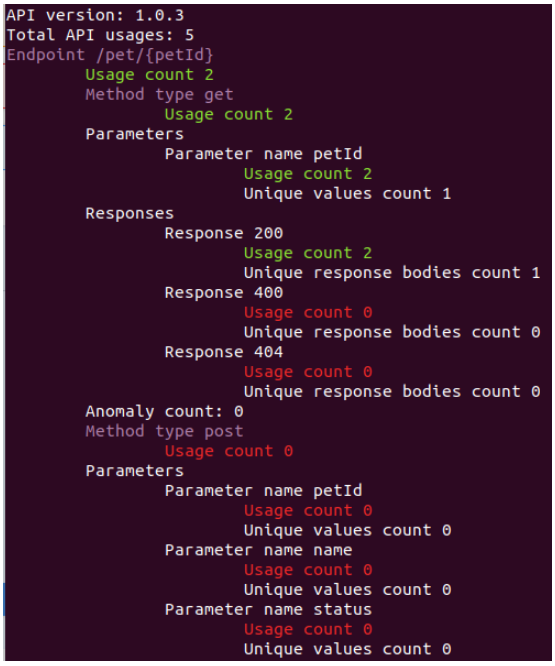
Edellä kuvatun käyttömäärien laskennan lisäksi jokaista verkkoliikenteen kirjausta verrataan rajapinnan kuvauksessa sitä vastaavaan määrittelyyn. ASC pyrkii havaitsemaan ja raportoimaan seuraavat rajapinnan kuvauksen ja toteutuneen verkkoliikenteen poikkeavuudet:

- Määrittelemätön HTTP-vastauskoodi. Mikäli HTTP-vastaus sisältää vastauskoodin, jota ei ole määritelty rajapinnan kuvauksessa kyseiselle päätepisteen operaatiolle, työkalu tulkitsee tilanteen poikkeamaksi. On mahdollista, että rajapinnassa on määritelty kyseiselle päätepisteen operaatiolle vakiovastaus, jonka on tarkoitus kattaa nämä tapaukset, joten työkalu luokittelee nämä tapaukset omiksi lajeikseen riippuen vakiovastauksen määrittelyn olemassaolosta rajapinnan kuvaustiedostossa.
- Puuttuva pakolliseksi määritelty HTTP-kutsun parametri. Rajapinnan kuvauksessa voidaan HTTP-kutsun sisältämät parametrit määritellä pakollisiksi. Mikäli tapahtuneesta HTTP-kutsusta ei löydy määriteltyjä pakollisia parametreja, tulkitsee työkalu poikkeaman tapahtuneen.
- Väärää muotoa oleva HTTP-kutsun sisältö. Mikäli HTTP-kutsun varsinainen sisältö ei vastaa rajapinnan kuvauksessa esitettyä mallia, tulkitsee työkalu tilanteen poikkeavaksi.
- Väärää muotoa oleva HTTP-vastauksen sisältö. Mikäli HTTP-vastauksen varsinainen sisältö ei vastaa rajapinnan kuvauksessa esitettyä mallia, tulkitsee työkalu tilanteen poikkeavaksi.
- Rajapinnan päätepisteen määrittelemättömän operaation kutsuminen. Mikäli HTTP-kutsu käyttää metodia, jota vastaavaa operaatiota ei ole määritelty kyseisessä rajapinnan päätepisteessä, tulkitsee työkalu tilanteen poikkeavaksi.
- Määrittelemätön HTTP-viestin sisällön mediatyyppi. Mikäli HTTP-kutsun tai -vastauksen sisällön mediatyypin määrittävät otsikkotiedot eivät vastaa rajapinnan kuvauksessa esitettyjä mediatyyppejä, tulkitsee työkalu kyseisen tilanteen poikkeamaksi.

Työkalun analysoitua kaiken verkkoliikenteen kirjoitetaan analyysin tulokset erilaisiin tiedostoihin. Havaitut poikkeukset ja yleisraportti analyysistä kirjoitetaan kumpikin omaan tekstitiedostoonsa. Yksityiskohtainen raportti kirjoitetaan myös JSON-tiedostomuotoiseen tiedostoon koneellista käsittelyä varten. Kuvassa 3 esitetään työkalun esimerkkitulostus suoritetusta analyysistä.

Valinnaisena parametrina voidaan työkaluun syöttää suorituksen alkaessa konfiguraatiotiedosto, joka säätelee työkalun toimintaa. Seuraavassa listauksessa käsitellään pinnallisesti konfiguraatiotiedoston sisältämät asetukset:

- Testauksen kattavuusehdot. Tämä asetus määrittää, millaista syvällistä kattavuutta testaukselta odotetaan. Mahdollisia asetuksia ovat täysi kattavuus rajapinnan päätepisteille, operaatioille, vastauksille tai parametreille. Rajapinnan parametrien kattavuusehdolle on määriteltävissä kaksi eri vaihtoehtoa: jokaista parametria on käytettävä vähintään joko kerran tai vähintään kaksi kertaa



Kuva 3. Esimerkki ASC:n komentoliittymään tulostamasta analyysistä

yksilöllisillä parametrin arvoilla. Näiden ehtojen täyttämättä jättäminen aiheuttaa ohjelman lopetuksen käyttämällä virhettä ilmaisevaa poistumiskoodia.

- Testauksen poikkeuksien esiintymisehdot. Tämä asetus määrittää listan poikkeuksista, joiden esiintyminen testauksessa aiheuttaa ohjelman lopetuksen käyttämällä virhettä ilmaisevaa poistumiskoodia.
- Muut työkalun toimintaan liittyvät asetukset, kuten haluttujen rajapinnan päätepisteiden huomiotta jättäminen tai rajapinnan palvelimen osoitteen tarkempi määrittäminen.
- Muut työkalun analyysin tulostukseen liittyvät asetukset, kuten loppuraporttien tiedostopolut ja niihin kirjoitettavan analyysin yksityiskohtien määrä.

Työkalun suorituksen lopussa viimeinen toiminto on tarkastaa, ovatko testaukselle asetetut kattavuus- ja poikkeuksien esiintymisehdot täyttyneet. Mikäli testauksen vaatimukset eivät täyty, työkalu lopettaa suorituksensa käyttämällä virheen ilmaisevaa poistumiskoodia. Tässä tapauksessa virheet testauksen kattavuudessa ja esiintyneet kielletyt poikkeavuudet testauksessa kirjoitetaan niille määriteltyihin omiin tiedostoihinsa. Nämä ominaisuudet mahdollistavat työkalun käyttämisen osana jatkuvan integraation järjestelmää, jossa työkalu varmistaa rajapintaa vastaan suoritettujen testien kattavuusvaatimusten täyttymisen.

5. TESTAUS

Tässä työn osiossa esitellään erilaisten automaattisten OpenAPI 2.0-spesifikaatiota (OAS 2.0) hyödyntävien rajapintojen testaustyökalujen tuottaman testauksen tarkastelu hyödyntämällä ASC:tä ja tässä osiossa esiteltävää testijärjestelyä.

Testattaviksi työkaluiksi valittiin viisi erilaista avoimen lähdekoodin työkalua, jotka kykenevät generoimaan ja suorittamaan automaattisesti joukon testitapauksia OAS 2.0:n mukaisesta rajapinnan kuvaustiedostosta.

Työkaluja testattiin kahdella erilaisella testillä: Ensimmäisessä testissä testaustyökalut asetettiin generoimaan testausta rajapintojen kuvaustiedostosta simuloitua palvelinta vastaan. Toisessa testissä käytettiin rajapinnan kuvaustiedoston mukaisia todellisen toiminnallisuuden sisältäviä kohteita testaustyökaluille. Molempien testien tarkoituksena oli selvittää testaustyökalujen tuottama testauksen kattavuus testikohteita vastaan. Toisessa testissä tarkoituksena oli selvittää testauksen kattavuuden lisäksi testauksen tuottamien rajapinnalta vastauksena saatujen rajapinnan kuvauksesta poikkeavien vastauskoodien määrät ja tyypit.

Molemmissa testeissä testityökalujen suorituksen aikainen verkkoliikenne tallennettiin kyseisiä testejä varten luodun testijärjestelyn avulla, ja testityökalujen tuottaman testauksen kattavuus suhteessa rajapinnan kuvaustiedostoon selvitettiin ASC:tä hyödyntämällä.

Testin kohteina olevien testaustyökalujen kohteina hyödynnettiin joukkoa OAS 2.0:n mukaisia rajapinnan kuvaustiedostoja, sekä joukkoa ohjelmistoja, jotka sisälsivät OAS 2.0:n mukaisen rajapinnan. Rajapintojen kuvaustiedostojen lähteenä käytettiin APIs.guru-palvelun[1] tarjoamaa julkista hakemistoa OAS:n mukaisista kuvaustiedostoista, josta valikoitiin 702 erilaista tämän käyttötarkoituksen vaatimukset täyttävää rajapinnan kuvaustiedostoa. Tämän lisäksi valittiin Github-versionhallintapalvelua[2] käyttämällä pieni joukko OAS 2.0:n mukaisen rajapinnan sisältäviä ohjelmistoja käytettäväksi testaustyökalujen testikohteina.

Testituloksissa esitetään työkalujen tuottaman testauksen kattavuus testimateriaalissa olleita kohteita vastaan käyttämällä erilaisia mittareita. Mittareina kattavuudelle käytettiin rajapintojen päätepisteiden, operaatioiden, parametrien ja vastauskoodien käytetyksi tuloa testauksen toimesta sekä mittaamalla, kuinka suuri osuus rajapinnoista, päätepisteistä ja operaatioista saivat testauksen toimesta täyden päätepisteiden, operaatioiden, parametrien tai vastauskoodien kattavuuden. ASC:n avulla myös testauksen aikana havaitut rajapinnassa määrittelemättömät HTTP-vastauskoodien ilmenemiset laskettiin ja tilastoitiin työkalu- ja rajapintakohtaisesti.

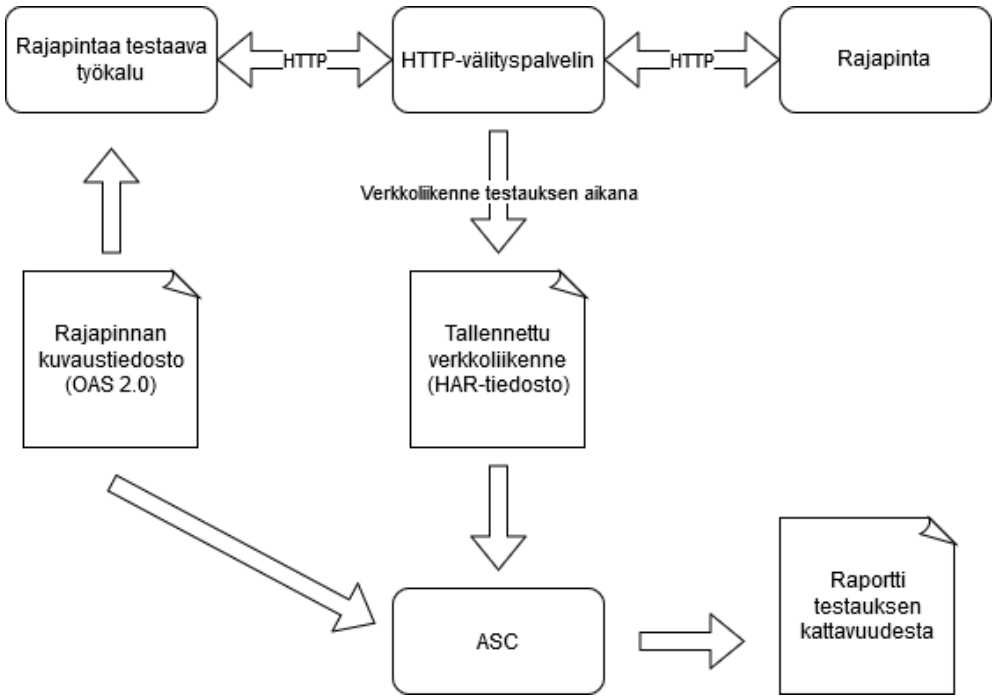
5.1. Testijärjestely

Kuva 4 havainnollistaa testijärjestelyn rakennetta. Testijärjestely koostuu neljästä komponentista, joita ovat testattavana oleva työkalu, työkalun testauksen kohteena oleva rajapinta, HTTP-välityspalvelin sekä ASC.

Testattavana olevalle työkalulle syötetään rajapinnan kuvaustiedosto ja muut tarvittavat tiedot, joiden perusteella työkalu generoi testisarjan ja aloittaa kyseisen

[1] https://apis.guru/openapi-directory/

[2] https://github.com/



Kuva 4. Kaaviokuva testijärjestelyn rakenteesta. Testauksen kohteena oleva työkalu testaa rajapintaa rajapinnan kuvaustiedoston perusteella. HTTP-välityspalvelin tallentaa verkkoliikenteen testien aikana. ASC analysoi ja raportoi testauksen kattavuuden rajapinnan kuvaustiedoston ja tallennetun verkkoliikenteen avulla

testisarjan suorittamisen. Työkalun toimintaan ei puututa millään tavalla suorituksen aikana.

HTTP-välityspalvelin tallentaa jokaisen HTTP-viestin testaustyökalun ja testattavana olevan rajapinnan välillä. Työkalun suorituksen loputtua HTTP-välityspalvelin kirjoittaa kaiken liikenteen HAR-tiedostoon myöhempää analyysiä varten.

Testauksen kohteena olevan työkalun suorituksen loputtua aloittaa ASC testauksen analysoinnin hyödyntämällä rajapinnan kuvaustiedostoa sekä HTTP-välityspalvelimen tallentamaa verkkoliikennettä.

5.2. Testien kuvaus

Tässä työssä suoritettujen testien tarkoitus oli testata testityökalujen toimintaa mittaamalla niiden suorittaman testauksen kattavuutta suhteessa testauksen kohteena olevan rajapinnan kuvaustiedostoon. Kattavuuden mittareina käytettiin rajapinnan kuvauksessa määriteltyjen elementtien käytetyiksi tulemista. Niitä työkaluja, jotka antoivat käyttäjälle mahdollisuuden säädellä automaattisesti generoitavien testitapausten määrää, testattiin usealla erilaisella luotavan testimäärän asetuksella.

Testauksessa suoritettiin kaksi erilaista testiä. Ensimmäisessä testissä testattiin työkalujen tuottamien HTTP-kutsujen kattavuutta rajapinnan kuvauksen suhteen käyttämällä testikohteena simuloitua palvelinta, joka ei sisältänyt rajapinnan

kuvaustiedoston mukaista toiminnallisuuden toteutusta. Toisessa testissä testattiin työkalujen tuottamien HTTP-kutsujen ja saatujen HTTP-vastausten kattavuutta käyttämällä testikohteena ohjelmistoja, jotka sisälsivät OAS 2.0:n mukaisen toiminnallisen rajapinnan toteutuksen. Toisessa testissä hyödynnettiin myös ASC:n kykyä etsiä verkkoliikenteestä poikkeamia suhteessa testauksen kohteena olevaan rajapinnan kuvaustiedostoon.

5.2.1. Testi 1: Testauksen HTTP-kutsujen kattavuus

Ensimmäisessä testissä tarkasteltiin työkalujen tuottaman testisarjan HTTP-kutsujen kattavuutta suhteessa rajapinnan kuvaustiedostoon. Jokainen työkalu suoritettiin jokaista testimateriaalissa olevaa rajapinnan kuvausta vastaan. Testikohteena toimi simuloitu palvelin, joka ei sisältänyt rajapinnan kuvaustiedostoa vastaavaa rajapinnan toteutusta. Koska simuloitu palvelin ei anna kyselyihin rajapinnan kuvausta vastaavia vastauksia, voidaan tässä testissä mitata ja analysoida ainoastaan testisarjan sisältämien HTTP-kutsujen kattavuutta.

5.2.2. Testi 2: Testauksen HTTP-kutsujen ja -vastausten kattavuus ja vastauskoodien poikkeukset

Toisessa testissä tarkasteltiin työkaluja todellisia kohteita käytettäessä. Testissä 2 tarkasteltiin työkalujen tuottamaa kattavuutta HTTP-kutsujen osalta samoin kuin testissä 1. Lisäksi testissä 2 analysoitiin testauksen saavuttamaa rajapinnan vastausten kattavuutta tarkkailemalla erilaisten HTTP-vastauksien esiintymistä testauksen aikana.

Testiajojen kattavuuden mittaamisen lisäksi suoritettiin ASC:n avulla testauksen aikana esiintyneiden HTTP-vastauskoodien vertailu rajapinnan kuvaustiedostoon mahdollisten rajapinnan kuvaustiedostossa määrittelemättömien HTTP-vastauskoodien löytämiseksi. Testauksen aikana löydetyt virheelliset vastauskoodit laskettiin työkalu- ja rajapintakohtaisesti.

5.3. Testattavat työkalut

Testattaviksi työkaluiksi valittiin viisi erilaista automaattiseen rajapinnan testaukseen kykenevää avoimen lähdekoodin työkalua. Kaikki valitut työkalut tukevat rajapinnan testauksen generoimista OAS 2.0:n mukaisesta rajapinnan kuvaustiedostosta.

Kaikki testiin valitut työkalut toimivat perusperiaatteeltaan samalla tavalla. Työkalu generoi testisarjan rajapinnan kuvaustiedoston ja mahdollisten muiden alkuparametrien perusteella, minkä jälkeen työkalu suorittaa generoidut testitapaukset rajapintaa vastaan HTTP-kutsuina ja tutkii saatujen HTTP-vastauksien tietoja raportoidakseen mahdolliset onnistumiset ja virheet.

- Schemathesis [32] on työkalu automaattiseen web-rajapintojen testaukseen. Työkalu tukee OAS:n (2.0 ja 3.0) mukaisia rajapintoja. Tässä työssä testattiin työkalun versiota 0.21.0, joka julkaistiin 20.12.2019. Työkalu

sisältää mahdollisuuden määritellä automaattisesti generoituvien testitapausten maksimimäärän yhdelle operaatiolle, mutta työkalu ei välttämättä aina hyödynnä täyttä annettua maksimimäärää testitapauksille. Työkalu tukee myös determinististä testitapausten generointia, jota myös hyödynnettiin tässä työssä esitettyjen testien suorittamisen aikana, jotta muutoksia työkalun eri suorituskertojen väleillä ei tapahtuisi. Työkalun pääkehittäjänä on toiminut Dmitry Dygalo.

- TnT-Fuzzer [29] on työkalu automaattiseen satunnaistestaukseen OAS 2.0:n mukaisille rajapinnoille. Työkalun alkuasetuksissa on mahdollista määrittää, kuinka monta kertaa jokaista työkalun tekemää HTTP-kutsua rajapinnalle iteroidaan ja lähetetään uudelleen testauksen aikana. Työkalun pääkehittäjinä ovat toimineet Timo Briddigkeit ja Tobias Hassenklöver. Työkalusta testattiin versiota 2.3.0, joka julkaistiin 10.9.2019.

- APIFuzzer [30] on työkalu automaattiseen satunnaistestaukseen. Työkalu tukee OAS 2.0:n mukaisia rajapintoja. Työkalusta julkaistiin versio 0.9 2.1.2018. Työssä testattiin työkalun erikseen numeroimatonta, 18.12.2019 GitHub-palvelusta noudettua, uusinta sovellusversiota.

- Meqa [31] on työkalu automaattiseen testien generointiin ja suorittamiseen OAS 2.0:n mukaisille rajapinnoille. Työkalun pääkehittäjänä on toiminut Ying Xie. Tässä työssä työkalusta testattiin versiota 0.66, joka julkaistiin 19.10.2017.

- “Got Swag?” [33] on työkalu, joka testaa OAS 2.0:n mukaisia rajapintoja automaattisesti satunnaistestauksella. Työssä käytettiin työkalun versiota 1.3.0, joka julkaistiin 21.2.2018. Työkalun pääkehittäjinä ovat toimineet Morris Brodersen ja Frederik Priede.

Työkaluja Schemathesis, TnT-Fuzzer ja APIFuzzer suoritettiin testien aikana Python 3.7 -komentotulkin avulla. Työkalua Meqa suoritettiin käyttöjärjestelmätason virtualisointia hyödyntävän Docker-ohjelmiston[3] avulla. Työkalu “Got Swag?” suoritettiin hyödyntäen Node.js-ympäristöä[4].

Myös muiden työkalujen valitsemista testikohteiksi harkittiin, mutta ne todettiin sopimattomiksi erilaisten syiden vuoksi:

- Dredd on työkalu automaattiseen Api Blueprint-kuvauskielen ja OAS 2.0:n mukaisten rajapintojen testaamiseen. Testauksen suorittaminen kuitenkin edellyttää, että rajapinnan kuvaustiedosto sisältää kaikille parametreille esimerkkiarvot, joita Dredd voi käyttää testaukseen. Tämän vaatimuksen vuoksi kyseinen työkalu todettiin epäsopivaksi testikohteeksi.

- Hamza Ed-douibi esitteli tutkimuksessaan luomansa mallin toteuttavan esimerkkityökalun OAS:n mukaisten rajapintojen automatisoituun testaukseen [38]. Luotu prototyyppi oli kuitenkin toimimattomassa tilassa siihen tehtävien parannuspäivitysten keskeneräisyyden vuoksi, joten kyseisen työkalun käyttäminen osana tätä työtä ei ollut mahdollista.

[3] https://www.docker.com/
[4] https://nodejs.org

- REST-ler [40] on täysin automaattinen, dynaamisesti ohjautuva työkalu OAS:n mukaisten rajapintojen testaukseen. Valitettavasti työkalua ei ole julkaistu, joten sen ottaminen testiin mukaan ei ollut mahdollista.

5.4. Testattavien työkalujen testimateriaali

Testiin 1 valittiin testattavien työkalujen testikohteiksi joukko OAS 2.0:n mukaisia rajapintojen kuvaustiedostoja. Testimateriaalin valinta suoritettiin rajapintojen hakemistoon APIs.guru rekisteröityjen rajapintojen joukosta.

Alustavasti testikohteiksi valittiin kaikki 1626 APIS.guru-palveluun rekisteröityä OAS:n mukaista rajapintaa ja niiden kuvaustiedostoa. Tästä 1626 kuvaustiedostoa sisältävästä joukosta karsittiin pois soveltumattomia kuvaustiedostoja seuraavin kriteerein:

- Rajapinnan kuvaustiedosto noudatti OAS:n versiota 3.0 halutun version 2.0 sijaan.
- Rajapinnan kuvaustiedosto tuotti virheilmoituksen käytettäessä rajapinnan kuvaustiedostoille tarkoitettuja validointi- tai jäsentämistyökaluja.
- Rajapinnan kuvaustiedosto ei sisältänyt yhtään päätepistettä.
- Rajapinnan kuvaustiedosto ei sisältänyt yhtään parametria.
- Rajapinnan kuvaustiedostosta löytyi rajapinnan päätepisteitä, joille ei oltu määritelty yhtään operaatiota.
- Rajapinnan kuvaustiedosto ei sisältänyt rajapinnan operaatioiden datatyyppien määritelmät kertovaa “definitions”-kenttää.
- Rajapinnan kuvaustiedostossa määritellyt rajapinnan päätepisteiden polut eivät olleet rajapinnan päätepisteet yksiselitteisesti yksilöivässä muodossa.
- Rajapinnan kuvaustiedosto ei koostunut yhdestä yksiselitteisestä tiedostosta, vaan sisälsi elementtiensä määrittelyissä viitteitä ulkopuolisiin tiedostoihin.
- Rajapinnan kuvaustiedoston määrittelemien rajapinnan päätepisteiden polut sisälsivät URL-merkkijonolle varattuja erikoismerkkejä, kuten &, ? tai #.
- Rajapinnan kuvaustiedosto sisälsi yli 20 määriteltyä päätepistettä tai 200 erikseen määriteltyä parametria.

Testikohteiden joukon karsinnan jälkeen jäi jäljelle 702 rajapinnan kuvaustiedostoa, joita käytettiin lopullisena testimateriaalina testissä 1. Kaikkiin kyseisen testin kuvaustiedostoihin lisättiin tai muokattiin rajapinnan verkkosijaintia määrittelevät “host”- ja “basePath”-kentät, jotta ne kertoisivat käytettäville testaustyökaluille asiaankuuluvasti testissä 1 käytetyn simuloidun palvelimen sijainnin. Testikohteina testissä 1 käytettyjen rajapintojen kuvaustiedostojen nimet ja palveluntarjoajat on listattu liitteessä 3.

Testiin 2 valittiin testattavien työkalujen kohteiksi julkisesti vapaasti saatavilla olevia ohjelmistoja, jotka sisältävät OAS 2.0:n mukaisen rajapinnan toteutuksen. Testikohteiden valitseminen suoritettiin etsimällä GitHub-versionhallintapalvelusta julkisia projekteja sopivilla hakusanoilla ja arvioimalla kyseisten projektien soveltuvuus testimateriaaliksi tapauskohtaisesti. Sopivien projektien etsimisen ja toimivuuden varmistamisen jälkeen kohteiksi valikoitui kuusi erilaista tähän testiin soveltuvaa projektia:

- Swagger Petstore[5]. Swagger Petstore on esimerkkisovellus Swagger-spesifikaatiota hyödyntävästä rajapinnasta.
- VideoGameDB[6]. VideoGameDB on James Willetin kehittämä esimerkkirajapinta, joka on tarkoitettu käytettäväksi esimerkkikohteena testityökalujen käyttökoulutuksessa.
- Prometheus Alertmanager[7]. Prometheus Alertmanager on palvelin, jonka tehtävä on vastaanottaa, käsitellä ja lähettää eteenpäin asiakasohjelmien lähettämiä hälytyksiä.
- Mailhog[8]. Mailhog on työkalu sähköpostipalvelimen testaukseen sekä lähtevän sähköpostin hallintaan.
- Jupyter Notebookserver[9]. Jupyter Notebookserver on palvelin Jupyter Notebook -verkkosovellukselle, joka mahdollistaa ohjelmakoodia sisältävien dokumenttien helpon luomisen ja jakamisen.
- ORY Oathkeeper[10]. ORY Oathkeeper on HTTP-kutsujen autentikointiin ja autorisointiin kehitetty ohjelmisto.

Kaikki testin 2 testikohteena käytetyt ohjelmistot asennettiin osaksi testijärjestelyä. Jokaisen testikohteen kuvaustiedoston “host”- ja “basePath”-kenttien arvoja muokattiin tarvittavalla tavalla, jotta kuvaustiedostot osoittaisivat testattaville työkaluille osaksi testijärjestelyä asennetun rajapinnan sijainnin.

5.5. Tulokset

Tässä osiossa esitellään testijärjestelyn tuottamat testitulokset. Testijärjestelyn mukaisen testin suorittaminen tuotti testauksen kattavuuden analyysin yhdellä työkalun suorituskerralla yksittäistä rajapintaa kohtaan. Kyseistä testijärjestelyä suoritettiin automatisoidusti eri rajapintojen ja työkalujen kanssa. Tämän jälkeen saadut tulokset yksittäisistä ajoista kerättiin ja yhdistettiin lopullisiksi tuloksiksi.

Testin 1 tuloksissa esitetään testaustyökalujen ajojen tuottamien testitapausten kattavuus rajapinnoille erilaisia mittareita käyttäen. Näitä mittareita olivat esimerkiksi

[5]http://petstore.swagger.io/
[6]https://github.com/james-willett/VideoGameDB
[7]https://github.com/prometheus/alertmanager
[8]https://github.com/mailhog/MailHog
[9]https://github.com/jupyter/notebook
[10]https://github.com/ory/oathkeeper

rajapinnan kuvauksessa esitettyjen elementtien, eli päätepisteiden, operaatioiden ja parametrien, käytetyksi tuleminen testauksen aikana. Yksittäisten elementtien käytetyksi tulemisen lisäksi tuloksissa esitetään työkalujen tuottama kattavuus rajapintojen, päätepisteiden ja operaatioiden näkökulmasta tarkastelemalla milloin niiden sisältämät erilaiset elementit ovat saavuttaneet täyden kattavuuden testauksen toimesta. Testin 1 tuloksista valittiin erillisen tarkastelun kohteeksi joukko rajapintoja, jotka tulivat onnistuneesti testatuksi kaikilla työkaluilla. Kyseistä tulosjoukkoa tarkasteltiin samoja mittareita käyttämällä vertailukelpoisempien tuloksien saamiseksi työkalujen välille.

Testin 2 tuloksissa esitetään päätepisteiden, operaatioiden ja parametrien katetuksi tuleminen samalla tavoin kuin testissä 1. Tämän lisäksi tuloksissa esitetään rajapinnan kuvauksessa määriteltyjen HTTP-vastauskoodien katetuksi tuleminen. Tuloksissa esitetään myös työkalujen suorituksen aikana havaitut rajapinnan määritelmästä poikenneet HTTP-vastauskoodit ja niiden määrät työkalu- ja rajapintakohtaisesti tilastoituna.

5.5.1. Tuloksissa käytetyt merkinnät ja termit

Taulukko 1 esittää lyhenteet ja lyhenteiden merkitykset, joita käytetään ilmaisemaan erilaisia suoritettuja työkalun ajoja. Esimerkiksi käytettävä lyhenne “TN 10” tarkoittaa työkalun TnT-Fuzzer suorittamista antaen työkalulle käskynä tuottaa 10 iteraatiota jokaiselle rajapintaan työkalun kohdistamalle testikutsulle.

Taulukko 1. Testien tuloksissa käytettyjen työkalujen ajojen lyhenteiden merkitykset

Lyhenne	Merkitys
SC [X]	Schemathesis, generoitavien testitapausten yläraja X
TN [X]	TnT-Fuzzer, käytettyjen iteraatioiden määrä X
AP	APIFuzzer
ME	Meqa
GO	Got Swag?

Tuloksien esityksissä käytetään termiä “onnistunut testaus”. Onnistuneeksi työkalun tuottamaksi testaukseksi tulkitaan työkalun yhtä rajapintaa testaava suoritus, joka täyttää seuraavat kolme ehtoa: 1) Työkalu ei kaadu sisäiseen virheeseen. 2) Luotu testaus ei ole tyhjä testijoukko. 3) Vähintään yksi luoduista testeistä kohdistuu kohteena olevan rajapinnan määriteltyyn operaatioon. Toisin sanoen, työkalun on suoritettava vähintään yksi rajapinnan operaatioon kohdistuva testitapaus ilman työkalun sisäistä virhettä.

5.5.2. Testin 1 tulokset

Testi 1 suoritettiin ajamalla työkaluja esitellyssä testijärjestelyssä rajapintojen kuvaustiedostojen kanssa käyttäen kohteena simuloitua palvelinta. Taulukko 2

esittää yleistiedot testin 1 sisältämistä testauksen kohteista ja suoritettujen testien yhteismäärästä.

Taulukko 2. Testin 1 yleiset tiedot

	Lukumäärä
Rajapintoja	702
Päätepisteitä	3549
Operaatioita	4885
Parametreja	21891
- Parametreja sijaintityypillä “path”	10112
- Parametreja sijaintityypillä “query”	9480
- Parametreja sijaintityypillä “header”	776
- Parametreja sijaintityypillä “body”	1182
- Parametreja sijaintityypillä “formData”	341
Testityökalujen suorituksia	10530
Onnistuneita työkalujen suorituksia	6078
Yksittäisiä testitapauksia	855379
Yksittäisiä testitapauksia onnistuneissa suorituksissa	781809

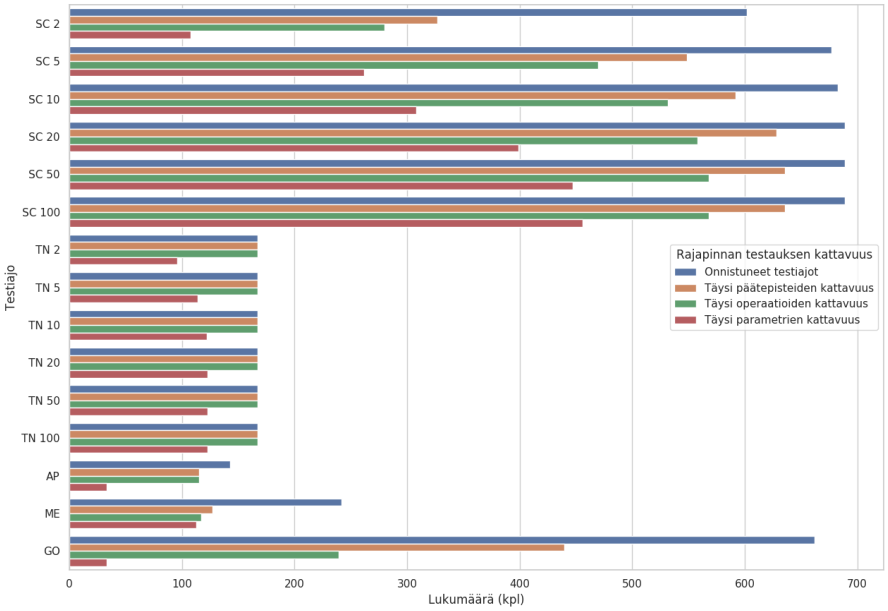
Työkalujen kyvyissä suorittaa testaus annetun rajapinnan kuvaustiedoston pohjalta esiintyi eroja. Työkalut Schemathesis ja “Got Swag?” tuottivat onnistuneen testauksen enemmistölle rajapinnoista, kun taas työkalut TnT-Fuzzer, APIFuzzer ja Meqa tuottivat onnistuneen testauksen vähemmistölle rajapinnoista. Tämän eroavaisuuden vuoksi tulokset esitetään kahdessa erilaisessa muodossa: ensin esitetään tulokset kaikkia testikohteita vastaan, minkä jälkeen esitetään testitulokset sille rajapintojen osajoukolle, jossa kaikki rajapinnat tulivat onnistuneesti testatuksi kaikkien työkalujen toimesta.

Kaikki rajapinnat

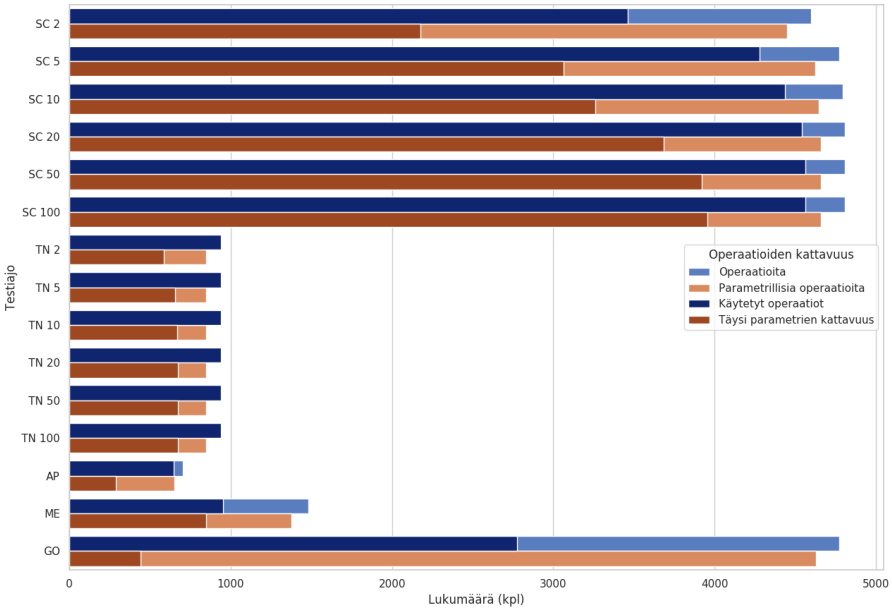
Kuva 5 esittää eri työkalujen suoritusten tuottaman kattavuuden kaikkien testijoukon rajapintojen näkökulmasta tarkasteltuna. Työkalut Schemathesis ja “Got Swag?” tuottivat onnistuneen testauksen enemmistölle kaikista rajapinnoista. TnT-Fuzzer tuotti testauksessaan täyden päätepisteiden ja operaatioiden kattavuuden kaikille onnistuneesti testaamilleen rajapinnoille.

Kuva 6 esittää eri työkalujen suoritusten tuottaman kattavuuden kaikkien testijoukon operaatioiden näkökulmasta tarkasteltuna. Schemathesis tuotti parhaimpia kattavuuden lukemia operaatioille, kun taas APIFuzzer jäi kattavuudessa heikoimmaksi. Päätepisteiden näkökulmasta kattavuuden lukemat noudattivat samaa linjaa: Schemathesis tuotti parhaimpia kattavuuden lukemia ja APIFuzzer heikoimpia. Tulokset päätepisteiden kattavuuden näkökulmasta esitetään liitteen 1 kuvassa 16.

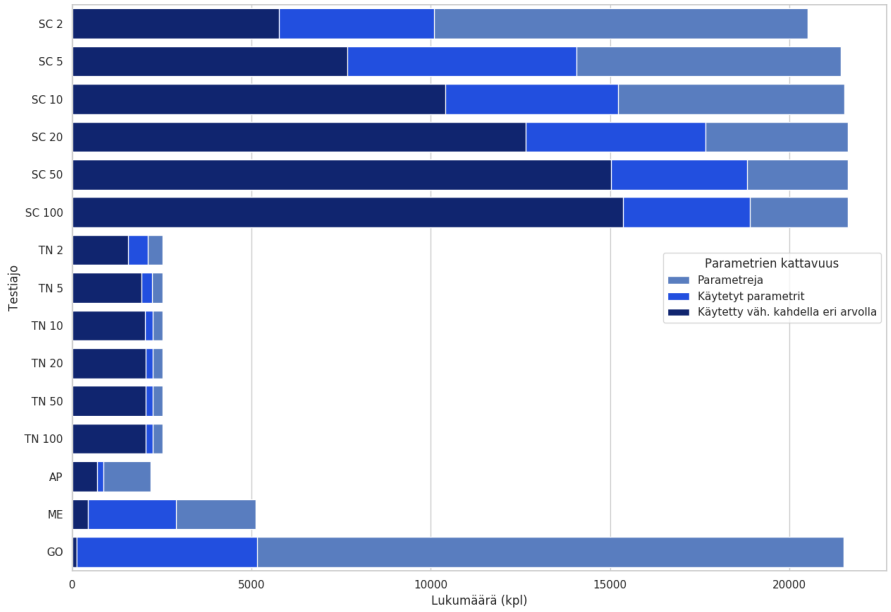
Kuva 7 esittää eri työkalujen suoritusten tuottaman kattavuuden kaikkien testijoukon parametrien näkökulmasta tarkasteltuna. Schemathesis tuotti suurimman



Kuva 5. Testi 1: Rajapintojen kattavuus



Kuva 6. Testi 1: Operaatioiden kattavuus



Kuva 7. Testi 1: Parametrien kattavuus

kokonaiskattavuuden parametreille ja sekä Schemathesis että TnT-Fuzzer testaavat enemmistöä testaamiensa rajapintojen parametreista kahdella erilaisella arvolla.

Parametrien kattavuutta tarkasteltiin myös parametrin sijaintityypin mukaisesti eriteltynä. Schemathesis kykeni testaamaan enemmistöä kaikista eri parametrityypeistä. Työkalujen TnT-Fuzzer ja Meqa suorittamassa testauksessa “header”-parametrit jäivät lähes kokonaan testaamatta. Työkalut “Got Swag?” ja APIFuzzer testasivat ainoastaan “path”-parametreja. Tulokset parametrien kattavuudelle sijaintityyppikohtaisesti esitetään liitteen 1 kuvissa 17 ja 18.

Kaikilla työkaluilla onnistuneesti testatut rajapinnat

Taulukko 3 esittää yleistiedot niistä testin 1 rajapinnoista, jotka olivat jokaisen työkalun onnistuneen testauksen kohteina. Rajatussa joukossa ei ollut yhtään “body”-parametria sisältävää rajapintaa, koska työkalu APIFuzzer ei kykene testaamaan niitä käyttäviä rajapintoja.

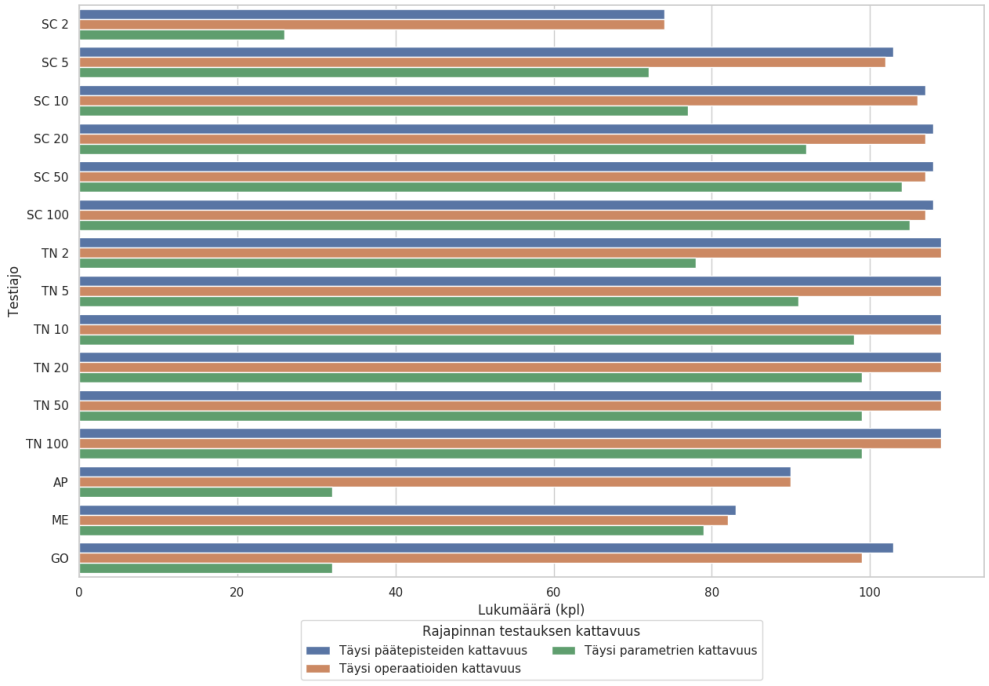
Yhteensä rajapintoihin kohdistuneita testitapauksia suoritettiin 292282 kappaletta, joiden tarkempi jakautuminen testiajojen välille esitetään liitteen 1 kuvassa 19. Työkalut “Got Swag?” ja Meqa generoivat alle 600 kappaletta testitapauksia, kun taas työkalu APIFuzzer generoi yli 90000 testitapausta. Työkalu TnT-Fuzzer kasvatti luomiensa testitapausten määrää suorassa suhteessa iteraatioasetukseensa nähden, kun taas työkalun Schemathesis tuottama testitapausten määrä ei ollut lineaarisessa suhteessa vastaavaan asetukseen.

Kuva 8 esittää eri työkalujen suoritusten tuottaman testauksen kattavuuden niiden rajapintojen näkökulmasta, jotka olivat jokaisella työkalulla onnistuneen testauksen kohteina. TnT-Fuzzer tuotti rajapinnoille täyden päätepisteiden ja operaatioiden kattavuuden, kun taas Schemathesis tuotti korkeammilla testitapaustensa määrillä korkeimman lukumäärän täysin parametreiltaan katettuja rajapintoja.

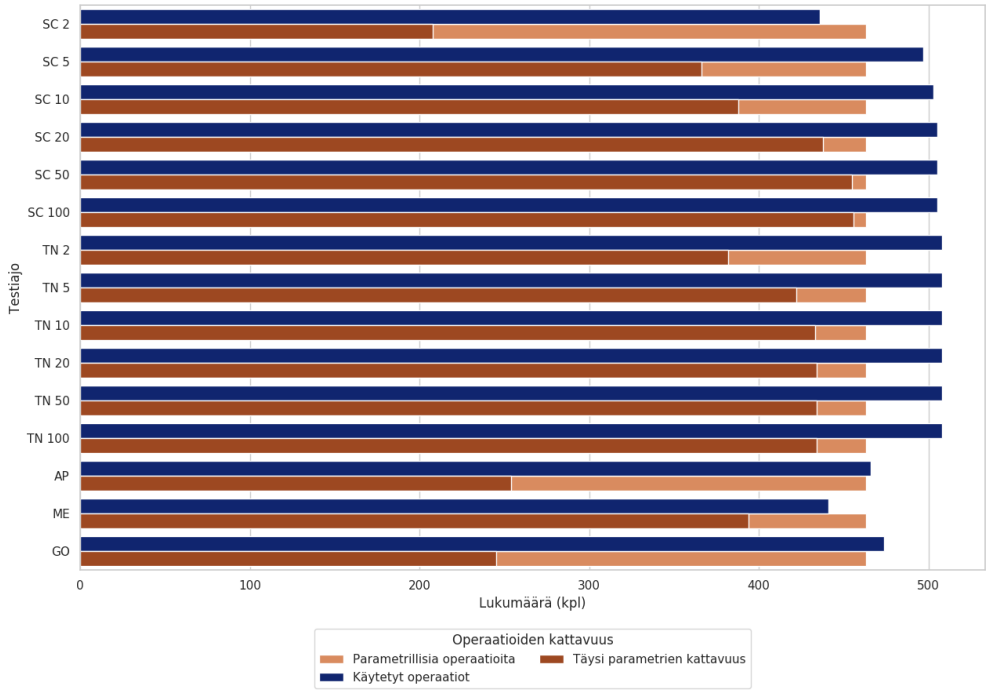
Taulukko 3. Testin 1 kaikilla työkaluilla onnistuneesti testatut rajapinnat

	Lukumäärä
Rajapintoja	109
Päätepisteitä	483
Operaatioita	508
Parametreja	1337
- Parametreja sijaintityypillä “path”	682
- Parametreja sijaintityypillä “query”	600
- Parametreja sijaintityypillä “header”	34
- Parametreja sijaintityypillä “body”	0
- Parametreja sijaintityypillä “formData”	21
Yksittäisiä testitapauksia	292282

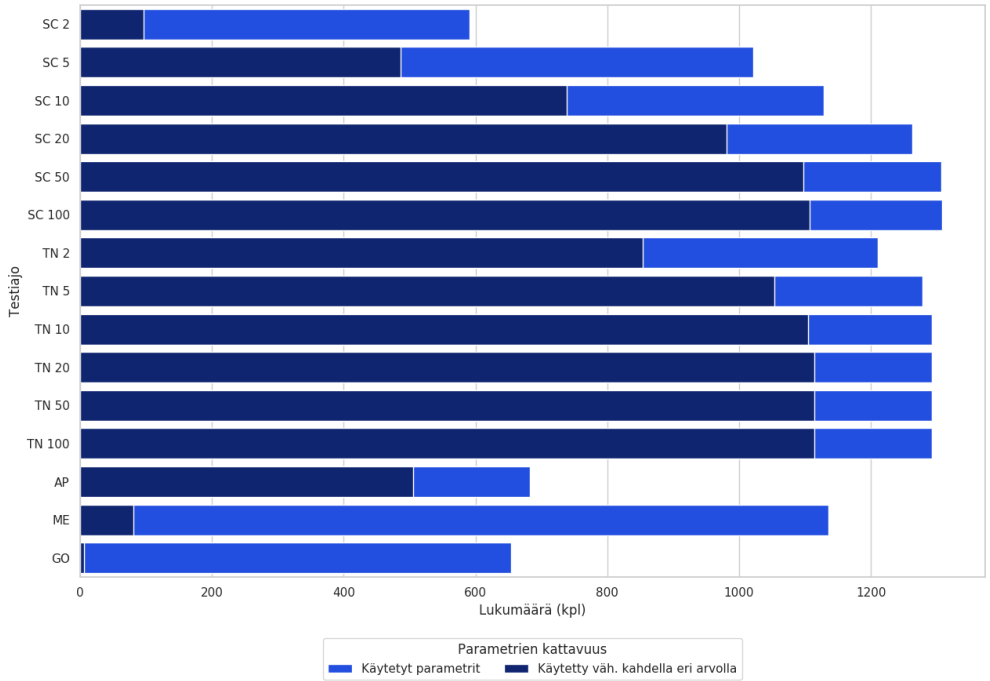
Kuva 9 esittää eri työkalujen suoritusten tuottaman testauksen kattavuuden operaatioiden näkökulmasta niissä rajapinnoissa, jotka olivat jokaisella työkalulla onnistuneen testauksen kohteina. Schemathesis ja TnT-Fuzzer saavuttivat korkeimmilla testitapausten määrillään parhaat kattavuuden tulokset. Työkalut APIFuzzer ja “Got Swag?” tuottivat matalimman määrän operaatioita, joissa kaikki parametrit ovat katettuja testauksen toimesta.



Kuva 8. Testi 1: Rajapintojen kattavuus, kaikilla työkaluilla onnistuneesti testatut rajapinnat



Kuva 9. Testi 1: Operaatioiden kattavuus, kaikilla työkaluilla onnistuneesti testatut rajapinnat



Kuva 10. Testi 1: Parametrien kattavuus, kaikilla työkaluilla onnistuneesti testatut rajapinnat

Päätepisteiden näkökulmasta saadaan samansuuntaisia tuloksia kuin operaatioiden näkökulmasta: Schemathesis ja TnT-Fuzzer saavuttivat parhaimmat kattavuuden tulokset, kun taas työkalut APIFuzzer ja “Got Swag?” tuottivat matalimman määrän päätepisteitä, joilla kaikki parametrit olivat katettuja testauksen toimesta. Tulokset päätepisteiden näkökulmasta kaikilla työkaluilla onnistuneesti testatuilla rajapinnoilla esitetään liitteen 1 kuvassa 20.

Kuva 10 esittää eri työkalujen suoritusten tuottaman kattavuuden parametrien näkökulmasta tarkasteltuna niissä rajapinnoissa, jotka olivat jokaisella työkalulla onnistuneen testauksen kohteina. Parhaan parametrien kattavuuden tuottivat työkalut Schemathesis ja TnT-Fuzzer, kun käytössä olivat korkeimmat asetukset luotavien testitapausten määrälle.

Parametrien kattavuutta tutkittiin myös parametrien sijaintityypin mukaan eriteltynä. Työkalu Schemathesis käytti kaikkia parametrityyppejä, kun taas APIFuzzer ja “Got Swag?” käyttivät ainoastaan “path”-parametreja. Tulokset eri sijaintityyppisten parametrien katetuksi tulemisesta testauksen aikana esitetään liitteen 1 kuvassa 21.

5.5.3. Testin 2 tulokset

Testi 2 suoritettiin ajamalla kaikkia työkaluja esitellyssä testijärjestelyssä käyttäen niiden testikohteina rajapinnan kuvaustiedoston toteuttavan toiminnallisuuden sisältäviä ohjelmistoja. Testin 2 tuloksissa esitetään tuotetun testauksen kattavuus rajapinnan, päätepisteiden, operaatioiden, parametrien ja vastausten näkökulmasta. Testin 2 tuloksissa esitetään myös ASC:n löytämät testauksen suorituksen aikana ilmenneet rajapinnan kuvaustiedostosta poikkeavat HTTP-vastauskoodit. Taulukko 4 esittää yleistiedot testin 2 sisältämistä testauksen kohteista ja testisuoritusten lukumäärästä. Liitteen 1 taulukko 9 esittää tarkemmat tiedot testissä 2 käytettyjen kohderajapintojen sisältämien päätepisteiden, operaatioiden, parametrien ja vastauskoodien määristä.

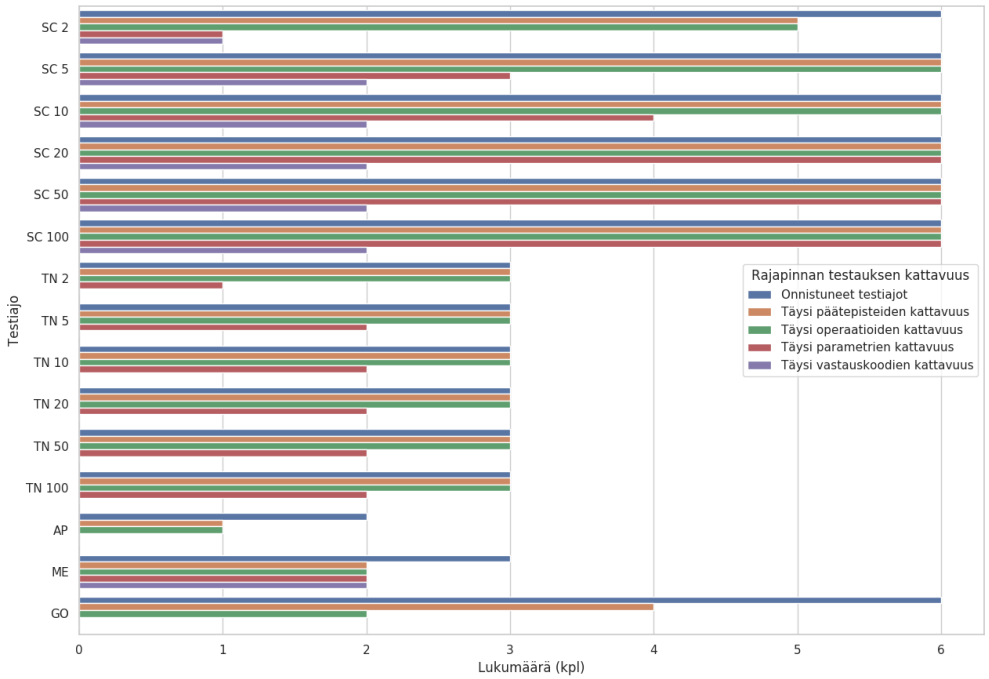
Kattavuus

Kuva 11 esittää testissä 2 mitatun työkalujen tuottaman testauksen kattavuuden rajapintojen näkökulmasta. Työkaluista Schemathesis tuotti kaikille testin rajapinnoille täyden päätepisteiden, operaatioiden ja parametrien kattavuuden käyttäessään korkeita asetuksia testitapausten luomismäärälle. Ainoastaan työkalut Schemathesis ja Meqa tuottivat kahdelle rajapinnalle täyden vastauskoodien kattavuuden, muut työkalut eivät saaneet vastaavaa tulosta yhdenkään rajapinnan testauksen osalta.

Kuva 12 esittää työkalujen suorituskohtaisen testauksen kattavuuden operaatioiden näkökulmasta. Täyden operaatioiden kattavuuden tuottivat onnistuneesti testaamilleen rajapinnoille työkalut Schemathesis ja TnT-Fuzzer. Työkalun Meqa tuottama testauksen kattavuus jäi yhtä operaatiota vajaaksi täydestä kattavuudesta. Päätepisteiden kattavuus noudatti tuloksissa samaa linjaa: Schemathesis ja TnT-Fuzzer saavuttivat täyden päätepisteiden kattavuuden testaamilleen rajapinnoille ja työkalu Meqa jäi yhtä päätepistettä vajaaksi täydestä kattavuudesta. Tulokset päätepisteiden kattavuuden näkökulmasta testissä 2 esitetään liitteen 1 kuvassa 26.

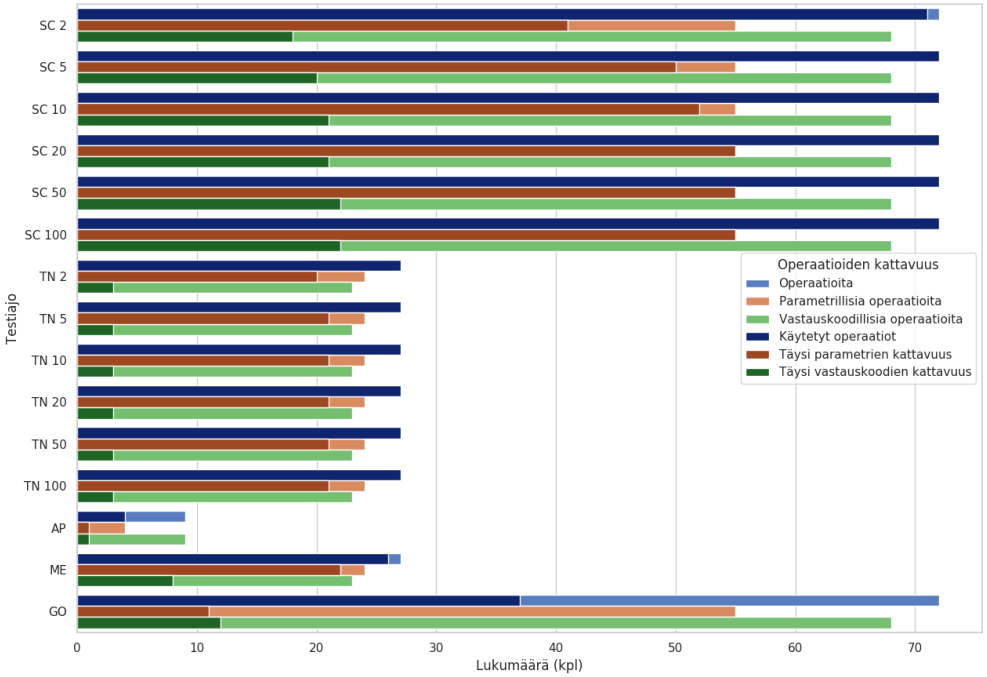
Taulukko 4. Testin 2 yleiset tiedot

	Lukumäärä
Rajapintoja	6
Päätepisteitä	46
Operaatioita	72
Parametreja	86
- Parametreja sijaintityypillä “path”	36
- Parametreja sijaintityypillä “query”	27
- Parametreja sijaintityypillä “header”	1
- Parametreja sijaintityypillä “body”	18
- Parametreja sijaintityypillä “formData”	4
Vastauskoodeja	131
Vakiovastauksia	4
Testityökalujen suorituksia	90
Onnistuneita työkalujen suorituksia	65
Yksittäisiä testitapauksia	23042
Yksittäisiä testitapauksia onnistuneissa suorituksissa	22475
Määrittelemättömiä HTTP-vastauskoodeja havaittu	66

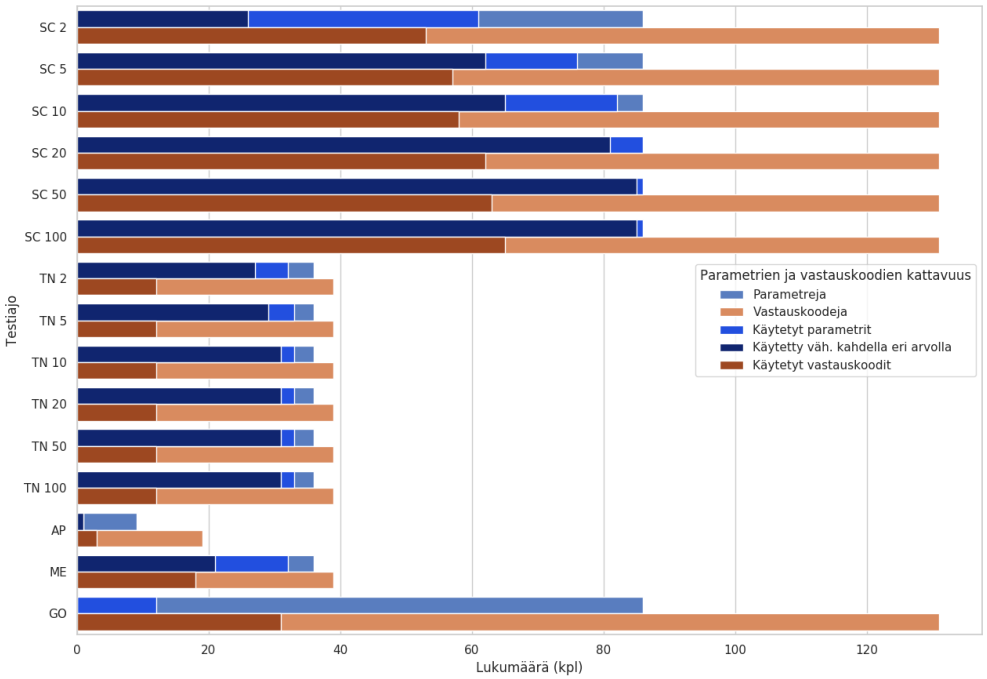


Kuva 11. Testi 2: Rajapintojen kattavuus

Kuva 13 esittää työkalujen suorituskohtaisen testauksen kattavuuden parametrien ja vastauskoodien näkökulmasta. Kaikkien testikohteiden osalta Schemathesis tuotti



Kuva 12. Testi 2: Operaatioiden kattavuus



Kuva 13. Testi 2: Parametrien ja vastauskoodien kattavuus

korkeimman vastauskoodien kattavuuden, ja siten suoriutui paremmin kuin vastaavia kohteita testannut työkalu “Got Swag?”. Samoja rajapintoja onnistuneesti testanneiden työkalujen TnT-Fuzzer ja Meqa keskinäinen vertailu osoittaa työkalun TnT-Fuzzer tuottaneen korkeamman parametrien kattavuuden ja työkalun Meqa tuottaneen paremman vastauskoodien kattavuuden.

Parametrien kattavuutta tarkasteltiin myös sijaintityyppikohtaisesti. Schemathesis testasi kaikkia parametrityyppejä. Työkalut TnT-Fuzzer ja Meqa testasivat kaikkia muita parametrityyppejä paitsi sijaintityypin “header”-parametria. Työkalut “Got Swag?” ja APIFuzzer jättivät testaamatta kaikkia muita parametreja paitsi “path”-parametreja. Tulokset parametrien kattavuuksista sijaintityyppikohtaisesti esitetään liitteen 1 kuvissa 27 ja 28.

Määrittelemättömät vastauskoodit

Työkalujen testisuoritusten saavuttaman kattavuuden lisäksi testissä 2 tarkasteltiin rajapinnan kuvauksessa määrittelemättömien HTTP-vastauskoodien esiintymistä testauksen aikana ASC:n avulla. Määrittelemättömäksi vastauskoodiksi tulkitaan sellainen operaation kuvauksessa määrittelemätön HTTP-vastauskoodi, joka saadaan vastauksena kyseistä operaatiota käyttävälle HTTP-kutsulle. Mikäli operaatiossa on määritelty vakiovastaus, tulkitaan kaikkien operaation testauksessa ilmenneiden erikseen määrittelemättömien vastauskoodien olevan vakiovastauksia, joten kyseisiä vastauskoodeja ei tulkita poikkeavuuksiksi.

Taulukko 5 esittää yksilöllisten määrittelemättömien vastauskoodien ilmenemisen työkalujen suoritusten aikana kutakin testauksen kohteena ollutta rajapintaa vastaan. Taulukossa esitetään myös jokaisen rajapinnan kaikki yksilölliset kaikkien työkalujen löytämät eri operaatioiden saamat määrittelemättömät vastauskoodit. Taulukossa käytettävä merkintä “X” tarkoittaa, että kyseisen työkalun suoritus ei ollut onnistunut. Schemathesis tuotti testauksellaan eniten rajapinnan kuvauksesta poikkeavia vastauskoodeja ja työkalu TnT-Fuzzer tuotti toiseksi eniten poikkeavuuksia. Rajapinnoissa “Swagger Petstore” ja “Jupyter Notebookserver” ilmeni yhteenlasketusti eniten yksilöllisiä vastauskoodien poikkeuksia kaikkien työkalujen testauksen aikana.

Taulukko 5. Testi 2: Yksilölliset määrittelemättömät vastauskoodit

	Swagger Petstore	VideoGameDB	Mailhog	Prometheus Alertmanager	ORY Oathkeeper	Jupyter Notebookserver
SC 2	10	1	1	3	1	22
SC 5	11	3	1	3	1	30
SC 10	11	5	1	4	1	31
SC 20	12	5	1	4	1	31
SC 50	14	5	1	4	1	32
SC 100	15	5	1	4	1	34
TN 2	6	4	1	X	X	X
TN 5	6	4	1	X	X	X
TN 10	7	4	1	X	X	X
TN 20	7	4	1	X	X	X
TN 50	8	4	1	X	X	X
TN 100	9	4	1	X	X	X
AP	X	X	1	X	1	X
ME	7	1	0	X	X	X
GO	0	1	1	1	0	2
Kaikki	18	8	1	4	1	34
Vastauskoodit	200, 404, 405, 415, 500	400, 404, 500	400	400, 404, 422	301	302, 400, 403, 404, 500

6. POHDINTA

Tässä työn osiossa analysoidaan tarkemmin suoritettujen testien tuloksia. Osiossa esitetään myös mahdolliset riskit testien luotettavuudelle ja pohditaan erilaisia tulevaisuuden kehityssuuntia suoritetuille testeille. Testien analysoinnin lisäksi tarkastellaan työssä esitellyn työkalun ASC merkittävyyttä testauksen arvioinnin kannalta sekä pohditaan työkalun tulevaisuuden kehitysmahdollisuuksia ja soveltuvuutta erilaisiin käyttötarkoituksiin.

6.1. Testitulosten analyysi

Tässä työssä suoritetuissa testeissä tarkasteltiin testaustyökalujen toimintaa tutkimalla tuotetun testauksen kattavuutta, työkalujen kykyä generoida testaus annetusta rajapinnan kuvaustiedostosta, työkalujen generoimien testien lukumäärää ja testauksen aikana esiintyneitä poikkeamia vastauskoodeissa.

Analyysissä vertailtiin eri työkalujen tuottaman testauksen ominaisuuksia ja vaikutuksia testikohteissa. Työkalun Schemathesis havaittiin tuottavan parhaan testauksen kattavuuden testikohteille useimmissa tapauksissa. Työkalu sai myös testikohteina käytetyistä rajapinnoista vastauksena eniten erilaisia poikkeavia vastauskoodeja.

Analyysissä vertailtiin työkalujen Schemathesis ja TnT-Fuzzer tarjoamien testitapausten määrää kontrolloivien asetuksien vaikutuksia tuotettuun testaukseen. Työkalut tuottivat korkeammalla luodulla määrällä testitapauksia parempia tuloksia testauksen kattavuuden ja poikkeuksien rajapinnasta löytämisen suhteen. Työkalun Schemathesis tuottaman testauksen tuloksien havaittiin vaihtelevan voimakkaammin riippuen annetuista asetuksista verrattuna työkalun TnT-Fuzzer tuottamaan testaukseen.

Saatujen tuloksien merkitystä pohdittiin myös käytännön ohjelmistotestauksen kannalta tarkastelemalla tuloksissa esiintyneitä työkalujen käytettävyyden näkökulmasta tärkeitä ominaisuuksia.

6.1.1. Testi 1

Testissä 1 tarkasteltiin työkalujen tuottamaa testauksen kattavuutta rajapintojen, operaatioiden ja parametrien näkökulmasta. Tässä osiossa analysoidaan saatuja tuloksia tarkemmin sekä tarkastellaan työkalujen Schemathesis ja TnT-Fuzzer käyttämien generoitujen testitapausten määrään vaikuttavien asetusten vaikutuksta.

Kaikki rajapinnat

Testissä 1 kattavuuden tulokset esitettiin kaikkien rajapintojen näkökulmasta. Johtuen työkalujen erilaisista kyvyistä tuottaa onnistuneeksi tulkittava testaus, eivät tulokset olleet työkalujen välillä suoraan vertailukelpoisia.

Testin 1 tiedot työkalujen tuottamista onnistuneiksi tulkituista testiajoista esitetään liitteen 1 taulukossa 6. Työkalujen kyvyissä generoida testiajoja havaittiin eroavaisuuksia:

- Työkalu Schemathesis tuotti 98%:lle rajapinnoista onnistuneen testauksen. Schemathesis epäonnistui testauksen tuottamisessa tapauksissa, joissa rajapinnan päätepisteet sisälsivät useita “path”-parametreja poluissaan. Näissä tapauksissa Schemathesis generoi ja suoritti testauksen asianmukaisesti, mutta yksikään testitapaus ei kohdistunut yhteenkään rajapinnan päätepisteeseen, jolloin testaus tulkittiin epäonnistuneeksi. Tämä ilmiö on havaittavissa käyttämällä matalia asetuksia luotavien testitapausten määrälle, jolloin päätepisteiden monimutkaisia parametreja sisältävät polut eivät saa asianmukaisia arvoja testauksen aikana.

- Työkalu “Got Swag?” tuotti onnistuneen testauksen 94%:lle rajapinnoista. “Got Swag?” käyttää luomassaan testauksessa ainoastaan HTTP:n GET-metodia, ja työkalu generoi tyhjän testijoukon (joka tulkitaan epäonnistuneeksi testaukseksi) jos rajapinnassa ei ole tällä metodilla käytettäviä operaatioita.

- Työkalut APIFuzzer, TnT-Fuzzer ja Meqa tuottivat onnistuneeksi tulkittavan testauksen vähemmistölle (eli 20%:lle, 23%:lle ja 34%:lle tässä järjestyksessä) kaikista rajapinnoista. Kyseisten työkalujen suoritukset pysähtyivät työkalujen sisäiseen virheeseen rajapintojen testauksen aikana.

Testissä 1 tarkasteltiin testauksen kattavuutta rajapintojen, päätepisteiden, operaatioiden ja parametrien näkökulmasta kaikille rajapinnoille.

- Parhaiten testauksen kattavuudessa päätepisteiden, operaatioiden ja parametrien suhteen kaikkia rajapintoja tarkasteltaessa menestyi työkalu Schemathesis. Schematheis kuitenkin epäonnistui testaamaan päätepisteitä, joissa polut sisältävät useita “path”-parametreja. Näissä tilanteissa myös kyseisten päätepisteiden sisältämät operaatiota ja parametrit jäivät testaamatta.

- TnT-Fuzzer onnistui tuottamaan täydellisen päätepisteiden ja operaatioiden kattavuuden kaikille onnistuneesti testaamilleen rajapinnoille. Vastaavaan suoritukseen ei pystynyt yksikään muu työkalu tässä testissä.

- Vaikka työkalu “Got Swag?” testasi ainoastaan HTTP:n GET-metodia, niin kyseisen metodin käytön yleisyyden vuoksi työkalu tuotti tässä tilanteessa kattavuuden enemmistölle kaikkien rajapintojen operaatioista.

Liitteen 1 kuvissa 22 ja 23 esitetään työkalujen Schemathesis ja TnT-Fuzzer tuottama kattavuus parametrien näkökulmasta suhteessa käytettyyn testien luontimäärän asetukseen. Vastaavat tulokset muille työkaluille esitetään liitteen 1 taulukossa 7. Molemmille työkaluille oli yhteistä parametrien kattavuuden paraneminen käytettäessä korkeampia luotavien testitapausten määriä:

- Schemathesis tuotti testauksellaan kasvavaa kattavuutta parametrien näkökulmasta käyttämällä korkeampia testitapausten määrän asetuksia työkalulle. Tuotetun kattavuuden kasvu alkoi tasaantumaan korkeimman asetuksen 100 käyttämisen kohdalla.

- TnT-Fuzzer tuotti parametrien näkökulmasta parhaan testauksen kattavuuden tason testien luontimäärän asetuksella 20. Asetuksen 10 jälkeen kattavuuden paraneminen oli enää marginaalista.

Työkaluilla onnistuneesti testatut rajapinnat

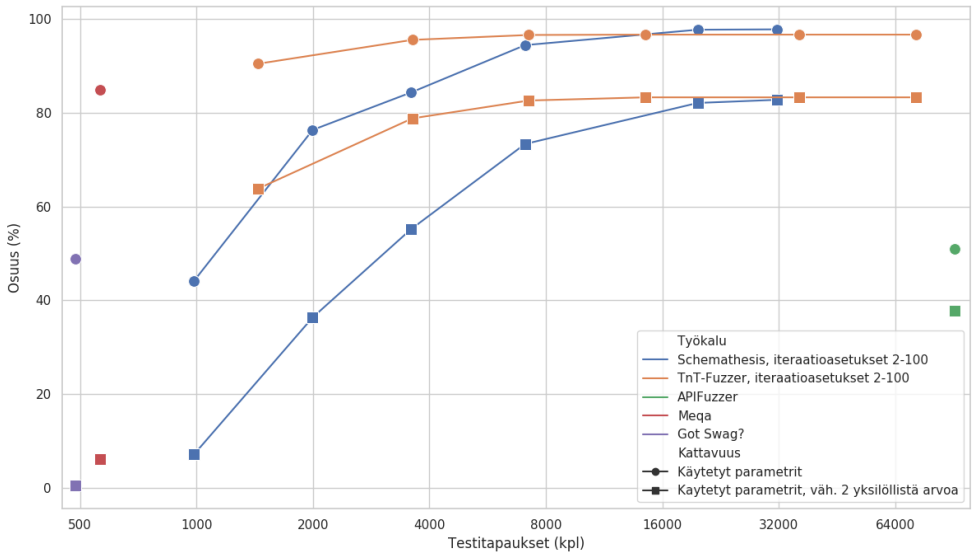
Testissä 1 tulokset esitettiin myös rajatulle joukolle rajapintoja, jotka tulivat onnistuneesti testatuksi kaikkien työkalujen toimesta. Supistettua joukkoa käyttämällä saadaan analysoitavaksi vertailukelpoisia tuloksia työkalujen välille rajapintojen, päätepisteiden, operaatioiden ja parametrien näkökulmasta. Päätepisteiden ja operaatioiden tulokset eivät vaihdelleet eri työkalujen välillä niin paljon kuin vastaavat parametrien tulokset vaihtelivat.

- TnT-Fuzzer tuotti testauksessaan täydellisen kattavuuden kaikille päätepisteille ja operaatioille. Parametrien kattavuudessa työkalu jäi marginaalisesti työkalua Schemathesis huonommaksi.
- Schemathesis tuotti supistetulle testikohteiden joukolle korkeimman parametrien kattavuuden. Päätepisteiden ja operaatioiden kattavuudessa työkalu jäi marginaalisesti työkalua TnT-Fuzzer huonommaksi.
- Meqa tuotti päätepisteille ja operaatioille työkaluista matalimman kattavuuden, mutta parametrien kattavuudessa työkalu saavutti paremman tuloksen kuin työkalut APIFuzzer ja “Got Swag?”.
- Työkalut APIFuzzer ja “Got Swag?” menestyivät heikoiten parametrien kattavuudessa.

Käytettävien testitapauksien luontimäärän asetuksen vaikutus työkalujen Schemathesis ja TnT-Fuzzer tuottaman testauksen saavuttamaan parametrien kattavuuteen rajatulla testikohteiden joukolla esitetään liitteen 1 kuvissa 24 ja 25 ja vastaavat parametrien kattavuuden lukemat työkaluille APIFuzzer, Meqa ja “Got Swag?” esitetään liitteen 1 taulukossa 8. Asetuksien vaikutukset rajatulla joukolla olivat samankaltaisia kuin täydellä joukolla:

- Schemathesiksen tuottaman testauksen kattavuuden kasvu osoitti hidastumisen merkkejä käytettäessä testitapausten määrän asetusta 100. Schemathesis suoriutuu matalimmalla asetuksellaan 2 heikosti, häviten kattavuudessa kaikille muille työkaluille.
- TnT-Fuzzer saavutti jo testitapausten luontimäärän asetuksella 20 tilanteen, jonka jälkeen kehitystä paremmaksi ei tapahtunut. Käytännössä asetuksen 10 jälkeen kattavuuden paraneminen oli enää marginaalista.

Työkalujen tuottama kattavuus suhteessa suoritettujen testitapausten määrään vaihteli eri työkalujen välillä. Kuva 14 esittää työkalujen tuottaman testauksen saavuttaman parametrien kattavuuden suhteessa kaikkien generoitujen testitapausten määrään rajatussa joukossa testikohteita.



Kuva 14. Testi 1: Työkalujen tuottama kattavuus parametreille suhteessa luotujen testitapausten määrään rajatussa testikohteiden joukossa

- Schemathesis saavutti parametrien kattavuudessa rajatun testijoukon parhaimman tuloksen käyttämällä testitapausten määrälle asetusta 50. Tätä asetusta käyttämällä työkalu generoi yhteensä 19743 testitapausta. Schemathesis jäi marginaalisesti työkalua TnT-Fuzzer heikommaksi vähintään kahdella yksilöllisellä arvolla käytettyjen parametrien määrän suhteen. Tulosten perusteella Schemathesis kykenee tuottamaan parametrien suhteen korkean kattavuuden testausta, mutta kattavuus riippuu voimakkaasti käytettävästä testitapausten määrän asetuksesta.

- TnT-Fuzzer tuotti testimäärän asetuksella 10 testauksensa kattavuuden suhteen tason, joka parani ainoastaan marginaalisesti käytettäessä suurempia asetuksia. Tämän kattavuuden saavuttamiseen vaadittiin 7230 testitapausta. Tämä on pienempi määrä testitapauksia kuin työkalulla Schemathesis, joka saavutti marginaalisesti paremman parametrien kattavuuden ja huonomman päätepisteiden, operaatioiden kattavuuden käyttämällä suurempaa määrää testitapauksia. Tulosten perusteella työkalu TnT-Fuzzer ei ole yhtä voimakkaasti riippuvainen käytetystä testitapausten määrän asetuksesta kuin työkalu Schemathesis.

- APIFuzzer ei menestynyt parametrien kattavuudessa huolimatta työkalun tuottamasta suurimmasta testitapauksien määrästä, joka oli yli 90000 kappaletta. Työkalun tuottama kattavuus jäi myös päätepisteiden ja operaatioiden suhteen toiseksi matalimmaksi. Enemmistö työkalun kattamista parametreista tuli testatuksi vähintään kahdella erilaisella arvolla. Tulosten perusteella testitapaukset keskittyivät vähemmistön parametreista testaukseen suurella joukolla satunnaisesti luotuja arvoja.

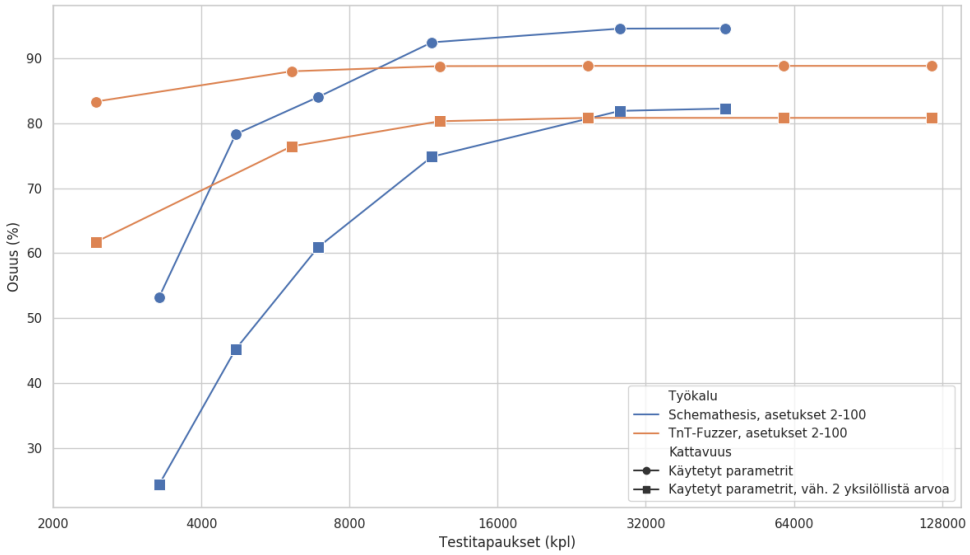
- Parametrien kattamisen osalta Meqa menestyi paremmin kuin työkalut APIFuzzer ja “Got Swag?” generoimalla rajatulle testijoukolle 564 testitapausta, joka on toiseksi alhaisin tuotettu määrä testitapauksia käytetyistä työkaluista.

- “Got Swag?” jäi heikoimmaksi kaikista työkaluista parametrien kattavuuden suhteen. “Got Swag?” tuotti 486 testitapausta kaikille rajapinnoille, joka on kaikkien työkalujen tuottamista testitapauksien määristä pienin. Verrattuna saman kokoluokan testitapausten määrän tuottaneeseen työkaluun Meqa, tuotti “Got Swag?” testauksellaan matalampaa parametrien kattavuutta ja vähemmän vähintään kaksi kertaa eri arvoilla testattuja parametreja. Tulosten perusteella työkalu “Got Swag?” tuotti kohteina olleille rajapinnoille minimalistisen testijoukon, jonka testauksen syvyys jäi parametrien osalta rajalliseksi.

Kuvassa 15 esitetään työkalujen Schemathesis ja TnT-Fuzzer keskinäinen vertailu parametrien kattavuudessa suhteessa luotujen testitapausten määrään kyseisillä työkaluilla onnistuneesti testattujen rajapintojen joukossa. Tulokset ovat samansuuntaisia kuin aikaisemmin esitetyt kyseisten työkalujen vastaavat vertailut:

- Schemathesis tuotti paremman kattavuuden korkeilla testitapausten määrillä. Edeltävistä vertailuista poiketen nousi Schemathesis paremmaksi myös parametrien katetuksi tulemisessa kahdella eri arvolla.

- TnT-Fuzzer tuotti paremman kattavuuden matalilla testitapausten määrillä.



Kuva 15. Testi 1: Työkalujen Schemathesis ja TnT-Fuzzer tuottama kattavuus parametreille suhteessa luotujen testitapausten määrään molemmilla työkaluilla onnistuneesti testatuiksi tulleiden rajapintojen joukossa

6.1.2. Testi 2

Testissä 2 tarkasteltiin työkalujen testauksen tuottamaa kattavuutta rajapintojen, päätepisteiden, operaatioiden, parametrien ja vastauskoodien näkökulmasta käyttämällä kohteina todellisen toiminnallisuuden sisältäviä rajapintoja. Myös testauksen tuottamat poikkeavat vastauskoodit esitettiin testin 2 tuloksissa.

Kattavuus

Testin 2 tulokset osoittavat työkalujen tuottamassa testauksessa eroavaisuuksia kattavuuden osalta:

- Schemathesis tuotti täydellisen kattavuuden kaikkien rajapintojen päätepisteille, operaatioille sekä parametreille. Työkalu tuotti myös parhaan kattavuuden rajapintojen vastauskoodeille.
- TnT-Fuzzer tuotti täydellisen päätepisteiden ja operaatioiden kattavuuden niille kolmelle rajapinnalle, joita työkalu testasi onnistuneesti. Työkalu kattoi testauksellaan enemmän parametreja kuin samoja rajapintoja onnistuneesti testannut työkalu Meqa.
- Meqa tuotti yhtä päätepistettä ja operaatiota vaille täydellisen kattavuuden kyseisille elementeille onnistuneesti testaamissaan rajapinnoissa. Kyseinen vajaus selittyy työkalun kykenemättömyydellä testata tiedoston lähettämistä odottavaa rajapinnan operaatiota. Vastauskoodien suhteen Meqa tuotti korkeamman kattavuuden kuin samoja rajapintoja testannut työkalu TnT-Fuzzer.
- “Got Swag?” testasi ainoastaan rajapintojen HTTP:n GET-metodia hyödyntäviä operaatioita sekä “path”-sijaintityypin parametreja. Tämä jätti parametrien kattavuuden alhaiseksi ja katettujen operaatioiden määrän puoleen kaikista operaatioista.
- Työkalun APIFuzzer tuottama testaus jäi kattavuudeltaan heikoimmaksi päätepisteiden, parametrien ja vastauskoodien suhteen. Samoin kuin työkalu “Got Swag?”, testasi APIFuzzer parametreista ainoastaan “path”-sijaintityypin parametreja.

Työkalujen Schematheis ja TnT-Fuzzer käyttäminen erilaisilla testitapausten määrää kontrolloivilla asetuksilla tuotti toisistaan poikkeavia vaikutuksia parametrien ja vastauskoodien kattavuuden osalta:

- Työkalulla Schemathesis testitapauksien määrää kasvattamalla saavutettiin täydellinen parametrien kattavuus kaikille testikohteille asetuksella 20 ja lähes täydellinen vähintään kahdella eri yksilöllisellä arvolla käytettyjen parametrien osuus asetuksella 50. Vastauskoodien osalta saavutettiin korkeampaa kattavuutta lisäämällä tuotettavien testitapausten määriä.

- Työkalulla TnT-Fuzzer eri testitapauksien määrän asetuksien käyttäminen tuotti vähäisiä vaikutuksia saavutettuun kattavuuteen. Parametrien kattavuus parani marginaalisesti asetukseen 10 asti ja vastauskoodien kattavuus säilyi samana riippumatta käytetystä asetuksesta.

Määrittelemättömät vastauskoodit

Testissä 2 tarkasteltiin työkalujen testauksen tuottaman kattavuuden lisäksi esiintyneiden vastauskoodien vastaavuutta rajapinnan kuvaustiedostoon ASC:n avulla.

Rajapinnan määrittelystä puuttuva vastauskoodi ei suoraan merkitse rajapinnan toteutuksen tai suunnittelun virhettä, sillä OAS sallii vastauskoodien määrittelemättä jättämisen. Kuitenkin kuvauskielen määritelmä toteaa, että rajapinnan kuvauksen “odotetaan määrittelevän vastaus onnistuneen operaation kutsumiselle ja jokaiselle tunnetulle virhetilanteelle” [20] [37]. Testaustyökalun voidaan katsoa onnistuneen tehtävässään, mikäli sen suorittaminen saa rajapinnalta esiin ennakoimattoman tilanteen määrittelemättömän vastauskoodin muodossa.

Rajapinnan vaillinaiset vastauskoodien määritelmät voidaan tulkita rajapinnan testattavuutta vähentäväksi ominaisuudeksi. Jos testaus generoidaan rajapinnan kuvaustiedoston perusteella, pakottaa vastausten puuttuminen kuvaustiedostosta testaajan tai työkalun arvaamaan rajapinnan odotettuja vastauksia suoritettaville testitapauksille.

Testauksen aikana ilmenneeksi yksilölliseksi tuntemattomaksi vastauskoodiksi tulkittiin operaation jokainen erilainen rajapinnan operaatiota käytettäessä ilmennyt määrittelemätön vastauskoodi. Poikkeava vastauskoodi koostuu yksilöllisestä operaation ja vastauskoodin yhdistelmästä ja useamman kertaan testauksen aikana ilmenneet samat vastauskoodit yhdelle operaatiolle tulkitaan tässä tarkastelussa yhdeksi poikkeamaksi. Mikäli operaatiolla oli määriteltynä vakiovastaus rajapinnan kuvauksessa, tulkittiin minkä tahansa vastauksen olevan määritelty.

Testissä 2 havaittiin työkalujen löytävän eri määriä poikkeavia vastauskoodeja testissä käytetyistä testikohteista:

- Schemathesis tuotti testauksellaan eniten poikkeavia vastauskoodeja.
- TnT-Fuzzer tuotti testaamilleen rajapinnoille toiseksi eniten poikkeavia vastauskoodeja. Rajapinnassa “Swagger Petstore” tuotti TnT-Fuzzer testauksellaan 3 kappaletta ja rajapinnassa VideoGameDB 2 kappaletta poikkeuksia, joita työkalu Schemathesis ei ollut löytänyt omalla testauksellaan. Muutoin kaikki löydetyt poikkeavat vastauskoodit sisältyivät työkalun Schemathesis löytämiin poikkeuksiin.
- Työkalut “Got Swag?”, Meqa ja APIFuzzer tuottivat vähäisemmän määrän poikkeuksia kuin työkalut Schemathesis ja TnT-Fuzzer, ja kaikki löydetyt poikkeukset sisältyivät näiden kahden työkalun löytämiin poikkeuksiin.

Työkalut Schemathesis ja TnT-Fuzzer menestyivät muita testin työkaluja paremmin poikkeuksien tuottamisessa. Selittävänä taustatekijänä tuloksille voidaan nähdä aikaisemmat testauksen havainnot, että molemmat työkalut käyttävät parametreja kattavammin kuin muut työkalut sekä luovat enemmän testitapauksia kuin esimerkiksi

työkalut “Got Swag?” ja Meqa. Esimerkiksi testattaessa rajapintaa “Swagger Petstore” tuotti Meqa 54 testitapausta ja TnT-Fuzzer tuotti 80-4000 testitapausta käytetystä luotavien testien määrän asetuksesta riippuen.

Työkaluilla Schemathesis ja TnT-Fuzzer luotujen testitapausten määrän kasvattaminen lisäsi löytyneiden poikkeavien vastauskoodien määrää. Rajapinnoissa VideoGameDB ja “ORY Oathkeeper” löytyneiden poikkeamien määrä ei kasvanut tai kasvu pysähtyi tietyn testitapausten luontimäärän asetuksen jälkeen. Rajapinnoissa “Swagger Petstore” ja “Jupyter Notebookserver” ei kuitenkaan saavutettu vastaavaa kasvun pysähtymistä poikkeaville vastauskoodeille. On mahdotonta tietää, onko jossakin testin rajapinnoista vielä löytämättä poikkeavia vastauskoodeja, mutta näiden tulosten valossa on syytä suorittaa lisätestausta rajapinnoille “Swagger Petstore” ja “Jupyter Notebookserver” käyttäen suurempaa luotavaa testitapauksien määrää.

Rajapinnoista löytyneiden poikkeavien vastauskoodien merkittävyys rajapinnan toiminnan tai rajapinnan kuvauksen oikeellisuuden kannalta vaihteli tapauskohtaisesti. Esimerkkinä poikkeuksien merkittävyydestä voidaan käyttää vastauskoodeja 200, 301 ja 302:

- Rajapinnassa “Swagger Petstore” havaittiin poikkeava HTTP-vastauskoodi 200 (operaation onnistuminen). Vastauskoodin puuttuminen operaation määritelmästä voidaan tulkita rajapinnan kuvauksen puutteellisuudeksi, koska operaation onnistunut suoritus pitäisi olla odotettu ja tunnettu lopputulos operaation käyttämisessä.

- Testauksen aikana havaittiin määrittelemättömiä HTTP-vastauskoodeja 301 ja 302, jotka ilmaisevat tarvetta uudelleenohjaukselle asiakkaan toimesta. Kyseiset poikkeukset aiheuttaneiden kutsujen tarkastelussa havaittiin niiden sisältävän merkkiin “/” päättyvän URL:n, minkä seurauksena palvelin palautti kehotuksen uudelleenohjaukselle samaan polkuun ilman sen päättävää “/”-merkkiä. Voidaan tulkita, ettei kyseisten poikkeuksien esiintyminen ilmaissut puutteita rajapinnan kuvauksen tai toiminnan kannalta.

6.1.3. Testitulosten merkitys työkalujen käytettävyydelle

Automaattisesti testejä luovien ja suorittavien testaustyökalujen tarkoitus on nopeuttaa ja helpottaa käytännön ohjelmistotestaustyötä siirtämällä kyseiset toimenpiteet pois ohjelmiston testauksen suorittajan vastuualueelta. Jos työkalu ei onnistu kyseisissä toiminnoissa tyydyttävällä tavalla, sen käyttökelpoisuus testaajan kannalta heikkenee.

Testaustyökalujen käytettävyyden näkökulmasta voidaan testituloksista nostaa esiin eri tekijöitä:

- Työkalun kyky prosessoida rajapinnan kuvaustiedosto ja luoda onnistuneeksi tulkittava testaus. Työkalusta, joka ei kykene luomaan testausta rajapinnan kuvaustiedoston perusteella tai jonka suoritus keskeytyy sisäiseen virheeseen testauksen aikana, ei ole käytännön hyötyä ohjelmistotestaajan näkökulmasta.

- Työkalun tuottaman testauksen kattavuus. Kattamattomat päätepisteet merkitsevät, että jotakin rajapinnan resurssia ei ole kokeiltu käyttää kertaakaan.

Kattamattomat operaatiot merkitsevät, että tiettyjä resurssien toiminnallisuuksia ei ikinä kokeiltu testauksen toimesta. Kattamattomat parametrit tarkoittavat, että on olemassa vielä kokeilemattomia tapoja tehdä kutsuja resurssien operaatiolle. Kattamattomat vastauskoodit tarkoittavat, että rajapinnalla on vielä olemassa mahdollisia lopputuloksia jonkin operaation käyttämiselle, joita ei testauksen toimesta saatu esiin. Mikäli työkalu ei testauksessaan kykene käyttämään tiettyjä parametrityyppejä, tulee niiden osalta testaus mahdottomaksi. Puutteet työkalun tuottaman testauksen kattavuudessa saattavat johtaa ohjelmistotestaajan näkökulmasta lisätyöhön tarvittavan kattavuuden saavuttamiseksi.

- Testitapauksien määrän kontrollointi. Testituloksien mukaan suuremmalla testitapauksien määrällä voidaan saavuttaa suurempaa kattavuutta ja enemmän rajapinnasta löytyviä poikkeuksia. Testitapauksien määrän säätelyn mahdollistavat työkalut antavat testaajalle kyvyn kontrolloida testauksen laajuutta testikohteen ja käytettävissä olevan aikamäärän mukaisesti.

Työkalu Schemathesis testasi onnistuneesti eniten rajapintoja, saavutti useimmissa tilanteissa parhaan testauksen kattavuuden ja työkalu sisältää mahdollisuuden kontrolloida testitapauksien määriä. Näiden seikkojen voidaan tulkita vaikuttavan positiivisesti työkalun käytettävyyteen verrattuna työssä tarkasteltuihin muihin neljään testaustyökaluun.

6.2. Riskit testitulosten oikeellisuudelle

Tässä työssä esitetyissä testituloksissa ja testien tuloksien analysoinnissa on tulosten oikeellisuuteen ja yleistettävyyteen vaikuttavia tekijöitä, jotka on otettava huomioon arvioitaessa saatujen tuloksien merkittävyyttä.

Testin 1 tuloksiin liittyi testikohteina käytettyjen rajapintojen suhteen riskitekijöitä:

- Testin 1 testimateriaalin valinnassa suoritettiin testikohteiden joukon rajaaminen alkuperäisestä 1626:sta kappaleesta alle puoleen esitettyjä kriteereitä käyttämällä. On mahdollista, että kriteerit rajasivat ulos tietyn tyyppisiä rajapintoja testikohteiden joukosta.

- Testin 1 rajapinnat olivat erikokoisia. Esimerkiksi 10 päätepistettä ja 20 operaatiota sisältävän rajapinnan onnistunut testaaminen saa testauksen kattavuuden mittarit näyttämään saman verran kyseisiä elementtejä katetuiksi kuin suurempi määrä pieniä, esimerkiksi vain 2 päätepistettä ja 2 operaatiota sisältäviä, rajapintoja. Samoin pienten ja yksinkertaisten rajapintojen täyden päätepisteiden, operaatioiden, parametrien tai vastausten kattavuuden saavuttaminen on keskimäärin helpompi tehtävä kuin suurien rajapintojen.

- Testissä 1 eriteltiin tulokset rajatulle joukolle kaikilla työkaluilla onnistuneesti testattuja rajapintoja. Rajattu joukko sisälsi 109 rajapintaa kaikkien testin 702:n rajapinnan joukosta. Työkalujen APIFuzzer, TnT-Fuzzer ja Meqa onnistuneiden suoritusten mukaiset rajapinnat korostuivat rajatun joukon tuloksissa. Rajatussa joukossa parametrin sijaintityypin “body” sisältävät

parametrit puuttuivat kokonaan, koska työkalu APIFuzzer ei testannut rajapintoja, joissa kyseistä parametrin sijaintityyppiä käytetään. Rajatussa testijoukossa yli 90% kaikista operaatioista käytti HTTP:n GET-metodia, joten saatuja tuloksia on tulkittava pääasiallisesti työkalujen kykynä testata kyseistä metodia hyödyntäviä operaatioita.

Testin 2 tuloksiin sisältyi riskejä sekä testikohteina käytettävien rajapintojen ominaispiirteiden että työkalujen suorittaman testauksen sattumanvaraisuuden vuoksi:

- Testissä 2 käytettiin testikohteina kuutta erilaista rajapintaa. Pienestä otannasta johtuen tuloksia voidaan pitää yleispätevien sijaan ainoastaan suuntaa antavina.
- Rajapinnat VideoGameBD ja Mailhog olivat kooltaan pienempiä kuin rajapinnat Swagger Petstore ja Jupyter Notebookserver, mikä antaa niille tehdylle testaukselle vähemmän painoarvoa saatuihin lopputuloksiin.
- Testissä 2 käytetyt rajapinnat Swagger Petstore ja VideoGameDB on suunniteltu ja toteutettu ainoana tarkoituksenaan tulla käytetyksi esimerkkinä OAS:n mukaisesta rajapinnasta tai testikohteeksi opetuskäyttöön. Muut kohteina käytetyt ohjelmistot ja niiden sisältämät rajapinnat olivat tarkoitettuja todelliseen käyttöön opetuskäytön sijasta. Vaikka ei ole syytä olettaa, että rajapinnoissa Swagger Petstore ja VideoGameDB olisi jotakin niiden tarkoitusperästä johtuvaa ominaispiirrettä, joka tekisi rajapinnoista vähemmän vertailukelpoisen muihin käytettyihin rajapintoihin, niin ei kyseistä mahdollisuutta voi sulkea pois.
- Työkalut TnT-Fuzzer, APIFuzzer ja “Got Swag?” tukeutuvat satunnaisten arvojen käyttöön luomissaan testitapauksissa, joten testaustuloksissa voi olla sen seurauksena vaihtelua. Tutkittaessa rajapintaan kohdistuvien kutsujen kattavuutta (eli päätepisteiden, operaatioiden ja parametrien kattavuutta) tämä ei muodostu ongelmaksi, koska kyseisten elementtien käytetyksi tuleminen on tärkeämpää kuin parametrien sisältämät satunnaiset arvot. Testissä 2 testikutsujen parametrien sisällöt kuitenkin vaikuttavat rajapinnan antamiin vastauksiin, mikä tuottaa satunnaisuutta testauksen tuottamaan vastauskoodien kattavuuteen. Testissä 2 joitakin vastauksia rajapinnalta ei ole mahdollista saada ilman tietyn tapahtumaketjun suorittamista, kuten esimerkiksi resurssin onnistunut luominen ja sitä seuraava resurssin muokkaaminen tai poistaminen, joten saatu vastausten kattavuus voi vaihdella testitapausten satunnaisuuden seurauksena.

6.3. Testien kehitysmahdollisuudet

Suoritetuille testeille on kehitysmahdollisuuksia, jotka toteutuessaan lisäisivät tulosten luotettavuutta ja yleispätevyyttä sekä toisivat uusia näkökulmia testien tulosten tarkastelulle.

Yksi kehityssuunta testien parantamisessa on lisätä työkalujen testikohteina käyttämien rajapintojen määrää, jotta saavutettaisiin suurempi määrä tietoa työkalujen toiminnasta. Testikohteiden määrän kasvattamiseen on olemassa useita tapoja:

- Testissä 1 hylättiin erilaisin perustein yli puolet alkuperäisestä käytettävien kohteiden joukosta, joka sisälsi 1626 erilaista rajapinnan kuvaustiedostoa. Osa hylätyistä rajapinnoista voidaan ottaa lisätoimenpiteiden avulla takaisin käyttökelpoiseksi testimateriaaliksi:
  - Yli 20 päätepistettä tai 200 parametria sisältävät rajapinnat jätetään karsimatta pois testeistä. Tämä kasvattaa testiajojen suorittamiseen kuluvaa aikaa sekä saattaa vaikuttaa tiettyihin suoritettuihin mittauksiin kuten tämän työn luvun testitulosten oikeellisuuden riskejä kuvaavassa osiossa on selitetty.
  - Rajapinnat, jotka sisälsivät osan määrittelyistään ulkopuolisessa lähteestä, kuten useaan osaan jaetusta rajapinnan kuvaustiedostosta, voidaan pyrkiä yhdistämään joko koneellisesti tai manuaalisesti yhdeksi rajapinnan kuvaustiedostoksi. Tätä takaisin yhdeksi koottua kuvaustiedostoa voidaan käyttää testimateriaalina testissä 1.
  - Rajapinnat, jotka tuottavat virheitä käytettäessä validointi- tai jäsentämistyökaluja, voivat olla korjattavissa selvittämällä virheiden syyt tapauskohtaisesti jokaisessa rajapinnan kuvaustiedostossa. Tämän jälkeen virheitä voidaan yrittää korjata muuttamalla kuvaustiedostoa kuitenkaan muuttamatta itse kuvauksen semanttista sisältöä, jolloin lopputuloksena saadaan testiin 1 soveltuva kuvaustiedosto.
- Kaikki testimateriaali testeihin hankittiin APIs.guru-, sekä GitHub-palveluista. Testimateriaalia voidaan lisätä etsimällä lisää rajapintoja testikohteiksi muista palveluista, kuten esimerkiksi palveluista ProgrammableWeb[1] tai RapidAPI[2]. Myös muita hakupalveluita, kuten Google[3], voidaan käyttää sopivien testikohteiden etsimiseen.
- Työkalujen APIFuzzer, TnT-Fuzzer ja Meqa kyky tuottaa onnistunut testaus kaikille testeissä käytetyille kohderajapinnoille jäi heikoksi. Jotta jokaisesta rajapinnasta saataisiin työkalujen tutkimisessa mahdollisimman suuri hyöty, voidaan jokaisen työkalun sisäiseen virheeseen päättynyttä suoritusta tutkia tarkemmin. Tämän jälkeen tutkitaan, voidaanko virhe välttää muokkaamalla rajapinnan kuvaustiedostoa muuttamatta sen semanttista sisältöä. Tämä vaihtoehto vaatii syvällistä tuntemusta rajapinnan kuvauskielestä, työkalujen sisäisestä toiminnasta ja manuaalista työtä.
- Testissä 2 testattiin rajapinnan sisältäviä ohjelmistoja, jotka oli mahdollista asentaa paikallisesti käytettäväksi testikohteeksi osaksi testijärjestelyä. Lisää testikohteita voidaan saada julkisesti saatavilla olevista rajapinnoista. Jokaisen uuden kohteen tekijöihin ja julkaisijoihin on otettava yhteyttä ja saatava lupa testauksen suorittamiselle, koska tuhansien (potentiaalisesti haitallisten) rajapintakutsujen suorittaminen saattaa vaikuttaa negatiivisesti rajapinnan toimintaan.

[1] https://www.programmableweb.com
[2] https://rapidapi.com/
[3] https://www.google.com/

Testeissä hyödynnettiin ASC:n kykyjä etsiä poikkeuksia rajapinnan kuvauksen ja testien aikaisen verkkoliikenteen välillä ainoastaan saatujen HTTP-vastauskoodien osalta. ASC:tä voidaan käyttää myös etsimään poikkeuksia HTTP-kutsuista ja siten varmistamaan, että testaus on käyttänyt rajapintaa myös tarkoituksellisesti muodoltaan rikkinäisillä tai puutteellisilla HTTP-kutsuilla.

Testattavien työkalujen määrää voidaan kasvattaa ottamalla mukaan tarkasteluun erilaisia periaatteita noudattavia testaustyökaluja. Esimerkiksi OAS:ta tukevat haavoittuvuustestaustyökalut, kuten Astra, OWASP Zed Attack Proxy ja Automatic API Attack Tool, voivat olla kiintoisia kohteita tarkastelulle. Muita testausperiaatteita hyödyntäviä työkaluja, kuten esimerkiksi rajapintojen lasilaatikkotestaukseen kykenevää työkalua EvoMaster[4], voitaisiin käyttää tutkimaan musta- ja lasilaatikkotestauksen eroavaisuuksia.

Testituloksia voidaan laajentaa mittaamalla kohteena olevien ohjelmistojen ohjelmistokoodin saavuttamaa kattavuutta testauksen aikana, mikäli kohteena käytettävä ohjelmisto on järkevästi instrumentoitavissa. Tällä tavalla voidaan avata mahdollisuus tutkia rajapinnan kattavuuden mittareiden, ohjelmistokoodin kattavuuden ja testaustyökalujen generoiman testauksen ominaisuuksien välisiä yhteyksiä.

Testien aikana ilmeni tilanteita, joissa työkalut Schemathesis ja TnT-Fuzzer olisivat saattaneet kyetä saavuttamaan paremman kattavuuden tai löytäneet enemmän poikkeuksia rajapintojen vastauskoodeista käyttämällä korkeampia testien luontimäärän asetuksia. Tämän mahdollisuuden selvittämiseksi voidaan työkaluja testata kyseisissä tilanteissa käyttämällä testitapausten luontimäärän asetuksena korkeampia arvoja, esimerkiksi 200, 400 tai 800.

6.4. Työkalun merkittävyys ja kehitysmahdollisuudet

ASC:n merkittävyyttä web-rajapintojen testauksen arvioinnille voidaan tarkastella työkalun nykyisten analyysiominaisuuksien, tulevaisuuden kehitysmahdollisuuksien ja työkalun erilaisiin käyttötarkoituksiin soveltuvuuden näkökulmasta.

ASC ei ole toiminnaltaan riippuvainen rajapinnan tai suoritetun testauksen toteutuksesta. Testauksen analysointiin vaaditaan rajapinnan kuvaustiedosto ja tallennettu verkkoliikenne, josta käyvät ilmi rajapinnalle tehdyt HTTP-kutsut ja -vastaukset. ASC voidaan nähdä itsenäisenä komponenttina, jota on mahdollista käyttää testauksen analysointiin ilman muutostarpeita testauksen suorittamiselle (lukuun ottamatta esimerkiksi mahdollista HTTP-välityspalvelimen käyttöä verkkoliikenteen tallentamiseksi) tai instrumentointia kohteena olevalle rajapinnalle. Yksinkertaisuutensa vuoksi voidaan ASC:tä hyödyntää osana muita järjestelmiä tai rakentamalla työkalun ympärille sen käyttöä sujuvoittavaa toiminnallisuutta. Esimerkki ASC:n hyödyntämisestä erilaisten työkalujen kehityksessä on liitteessä 2 esitetty sovellus (Arora, Hilke, Moberg, Männikkö ja Säärelä). Sovellus hallinnoi verkkoliikennettä tallentavan välityspalvelimen toimintaa, syöttää automaattisesti tarvittavat tiedot ASC:lle ja näyttää käyttäjälle testauksen analyysin graafisen käyttöliittymän avulla.

[4] https://github.com/EMResearch/EvoMaster

ASC:n ominaisuudet OAS:n mukaisen rajapinnan testauksen analysointiin eivät ole ennenkuulumattomia: Työkalu SoapUI kykenee laskemaan suorittamansa testauksen kattavuuden suhteessa rajapinnan kuvaustiedostoon. Työkalu openapi-cop[5] (julkaistu 1.3.2020) on välityspalvelin, joka tarkastelee verkkoliikenteen kutsujen ja vastauksien vastaavuutta annettuun OAS:n mukaiseen kuvaustiedostoon.

ASC arvioi rajapinnalle suoritettua testausta rajapinnan kuvauksessa esiintyvien elementtien kattavuuden näkökulmasta. Kyseisessä testauksen kattavuuden tarkastelussa ilmeinen ongelma on, että kattavuus kertoo ainoastaan puutteet tarkasteltavan kohteen osalta. Vaikka esimerkiksi operaatio tai parametri olisi tullut katetuksi, se ei varmista testauksen olleen tältä osin riittävää. Vastauskoodien kattavuus on samalla tavoin hyödyllinen ja rajallinen puutteiden osoittaja. Kuitenkin vastauskoodien osalta OAS sallii niiden määrittelemättä jättämisen tai todellisuudessa mahdottomien vastauskoodien määrittelyn rajapinnalle, joten todenmukaisten tulosten saaminen on riippuvaista rajapinnan määritelmän oikeellisuudesta.

ASC sisältää useita mahdollisuuksia poikkeuksien etsinnöille. Suoritetuissa testeissä hyödynnettiin poikkeavien vastauskoodien etsintää, joka osoittautui hyödylliseksi tavaksi paljastamaan rajapinnan kuvauksen puutteita. Poikkeamat on kuitenkin tutkittava tapauskohtaisesti niiden todellisen merkittävyyden selvittämiseksi. ASC sisältää esimerkiksi myös ominaisuuksia HTTP-vastausten sisällön ja mediatyyppien vertailuun rajapinnan kuvaustiedoston suhteen. Nämä ominaisuudet voisivat olla hyödyksi rajapinnan toiminnan ja kuvaustiedoston yhtenevyyden varmistamisessa silloin, kun testaus ei itsessään varmista saatujen vastausten muodon oikeellisuutta.

ASC:n tulevaisuuden kehitysmahdollisuudet liittyvät työkalun nykyisten ominaisuuksien parantamiseen:

- HTTP-kutsujen ja -vastauksien viestirunkojen hienostuneempi purkaminen ja analysointi. Työkalua voidaan kehittää purkamaan kutsujen ja vastausten mahdollisesti sisältämät JSON-muotoiset objektit yksittäisten kenttien tasolle, mikä mahdollistaa kyseisten objektien tarkemman analysoinnin.

- Työkalun rajapintojen kuvauskielien tuen laajentaminen. Rajapintojen kuvauskielet RAML ja API Blueprint ovat perusrakenteeltaan samankaltaisia kuin OAS. Molemmat sisältävät tiedot rajapinnan päätepisteistä, käytettävistä HTTP-metodeista sekä HTTP-kutsujen ja HTTP-vastausten sisällöistä, joten ASC:n tuen laajentaminen näille kuvauskielille on mahdollista.

- Työkalun suorittaman kattavuuden tarkastelun laajentaminen. ASC tukee tällä hetkellä yksinkertaistettua kattavuuden tarkastelua rajapinnan päätepisteiden, operaatioiden, parametrien ja vastausten osalta. Kattavuuden tarkastelua on mahdollista laajentaa Martin-Lopez et al. tutkimuksessaan [39] esittämän kattavuuden mallin suuntaan tarkemman testauksen analyysin suorittamiseksi.

- Laajemmat mahdollisuudet työkalun käyttäjälle määritellä erilaisia kattavuuden ja poikkeuksien esiintymisehtoja. Tarjoamalla monipuolisemmat mahdollisuudet käyttäjälle määritellä ehtoja tilanteille, joissa työkalu lopettaa suorituksensa käyttämällä virhettä ilmaisevaa poistumiskoodia, saadaan työkalu soveltumaan paremmin toimimaan osana jatkuvan integraation järjestelmää.

[5] https://github.com/EXXETA/openapi-cop

7. YHTEENVETO

Ohjelmistojen testaus on tärkeä osa ohjelmistokehitystä. Uusien testausmenetelmien ja testaustyökalujen kehittäminen ohjelmistotestauksen nopeuttamiseksi, helpottamiseksi ja testauksen perinpohjaisuuden parantamiseksi on kiintoisa kohde liiketoiminnalle ja tutkimukselle. Tapojen ja työkalujen luominen ohjelmistotestauksen ominaisuuksien arviointiin tarjoaa arvokasta tietoa ohjelmistotestauksen suorittajille testauksen aikana ja testauksen arvioinnin kehittymisen voidaan myös nähdä olevan pidemmällä ajanjaksolla ohjelmistotestausta ja ohjelmistotestaustyökalujen kehitystä eteenpäin auttava voima.

Tässä työssä esiteltiin työkalu API Specification Coverage tool (ASC) OpenAPI-spesifikaation mukaisten rajapintojen testauksen analysointiin. ASC laskee rajapinnan testauksen kattavuuden rajapinnan kuvaustiedoston ja tallennetun verkkoliikenteen avulla. Työkalu etsii verkkoliikenteestä poikkeavuuksia suhteessa rajapinnan kuvaustiedostoon paljastaakseen virheet testeissä, rajapinnan kuvauksessa tai rajapinnan toteutuksessa. ASC voidaan liittää osaksi jatkuvan integraation järjestelmää haluttujen testauksen kriteerien automaattiseksi varmistamiseksi. ASC on yksinkertaisuutensa ansiosta liitettävissä muihin järjestelmiin tai sovelluksiin, jotka voivat tarjota käyttöä helpottavia lisäominaisuuksia työkalun ympärille. ASC:n tulevaisuuden kehitysmahdollisuudet keskittyvät nykyisten ominaisuuksien jatkokehitykseen: tuki erilaisille rajapintojen kuvauskielille, yksityiskohtaisempi HTTP-viestien tarkastelu, kattavuuden tarkastelun ja jatkuvan integraation järjestelmien yhteensopivuuden jatkokehittäminen ovat potentiaalisia kehityssuuntia työkalun hyödyllisyyden lisäämiseksi.

Työssä tutkittiin ASC:n avulla viittä erilaista testaustyökalua (Schemathesis, TnT-Fuzzer, APIFuzzer, Meqa ja “Got Swag?”), jotka kykenivät luomaan automaattisesti testauksen OAS:n mukaisille rajapinnoille. Työkalujen testikohteiksi valittiin 702 rajapinnan kuvaustiedostoa ja 6 todellista rajapintaa. Työkalujen tuottamaa testausta rajapinnoille tarkasteltiin rajapinnan kuvaustiedostossa määriteltyjen päätepisteiden, operaatioiden, parametrien ja vastauskoodien katetuksi tulemisen ja testauksen aikana ilmenneiden rajapintojen kuvauksista poikkeavien HTTP-vastauskoodien suhteen. Työkaluja Schemathesis ja TnT-Fuzzer, joiden ominaisuudet tukivat vaihtelevan määrän testitapauksien luomista, testattiin useilla erilaisilla generoitavien testitapauksien määrillä.

Testeissä työkalu Schemathesis tuotti useimmissa tilanteissa parhaimman testauksen kattavuuden sekä korkeimmat määrät rajapintojen kuvauksista poikkeavia vastauksia. Testitapauksen määriä kontrolloivilla työkaluilla Schemathesis ja TnT-Fuzzer havaittiin generoitavien testitapausten määrän kasvattamisen tuottavan parempia tuloksia. Generoitavien testitapausten määrän asetuksen käyttö eri arvoilla tuotti suurempia vaihteluita työkalun Schemathesis tuottamaan testaukseen kuin työkalun TnT-Fuzzer tuottamaan testaukseen.

Suoritetuille testeille on olemassa useita jatkokehitysmahdollisuuksia. Potentiaaliset jatkokehityksen suunnat suoritetulle testaukselle sisältävät testauksen kohteina käytettävien rajapintojen ja tarkastelun kohteina olevien työkalujen määrän kasvattamisen erilaisilla tavoilla. Uusien tapojen kehittäminen testauksen arviointiin esimerkiksi testikohteena käytetyn rajapinnan instrumentoinnin avulla sisältyvät myös mahdollisiin kehityssuuntiin.

8. VIITTEET

[1] Measuring digital development facts and figures 2019 (2019). International Telecommunication Union. URL: https://www.itu.int/en/ITU-D/Statistics/Documents/facts/FactsFigures2019.pdf. Noudettu: 31.3.2020.

[2] Santos W. (2019), Apis show faster growth rate in 2019 than previous years. ProgrammableWeb. URL: https://www.programmableweb.com/news/apis-show-faster-growth-rate-2019-previous-years/research/2019/07/17. Noudettu: 31.3.2020.

[3] Santos W. (2017), Which api types and architectural styles are most used? ProgrammableWeb. URL: https://www.programmableweb.com/news/which-api-types-and-architectural-styles-are-most-used/research/2017/11/26. Noudettu: 24.9.2019.

[4] Garret R., Swizter S., Amundsen M., Medjaoui M. & Lascelles F. (2019), The state of api integration 2019 report. Cloud Elements. URL:https://offers.cloud-elements.com/the-state-of-api-integration-2019. Noudettu: 25.9.2019.

[5] Fielding R. (2000) Architectural styles and the design of network-based software architectures. Väitöskirja, University of California.

[6] Wagner J., Understanding the api-first approach to building products. SmartBear Software. URL: https://swagger.io/resources/articles/adopting-an-api-first-approach/. Noudettu: 7.10.2019.

[7] Rosenstock L. (2018), Openapi and design-first principles. Stoplight. URL: https://stoplight.io/blog/openapi-and-design-first-principles-96e7c4b2aec1/. Noudettu: 7.10.2019.

[8] Sandoval K. (2017), Using a schema-first design as your single source of truth. Nordic APIs AB. URL: https://nordicapis.com/using-a-schema-first-design-as-your-single-source-of-truth/. Noudettu: 7.10.2019.

[9] The 2019 state of api report (2019). SmartBear Software. URL: https://smartbear.com/SmartBearBrand/media/pdf/SmartBear_State_of_API_2019.pdf. Noudettu: 25.9.2019.

[10] Grigorik I. (2013) High Performance Browser Networking. O'Reilly Media, Inc.

[11] Fielding R.T. (2008), Rest apis must be hypertext-driven. Untangled. URL: https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven. Noudettu: 24.9.2019.

[12] Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P. & Berners-Lee T. (1999), Rfc 2616: Hypertext transfer protocol–http/1.1. Network Working

Group. URL: https://tools.ietf.org/html/rfc2616. Noudettu: 5.10.2019.

[13] Berners-Lee T., Fielding R. & Masinter L. (2005), Uniform resource identifier (uri): Generic syntax. Network Working Group. URL: https://www.ietf.org/rfc/rfc3986.txt. Noudettu: 6.10.2019.

[14] Roy Fielding J.R. (2014), Rfc 7231-hypertext transfer protocol (http/1.1): Semantics and content. Internet Engineering Task Force (IETF). URL: https://tools.ietf.org/html/rfc7231. Noudettu: 4.10.2019.

[15] Dusseault L. & Snell J. (2010), Patch method for http. Internet Engineering Task Force (IETF). URL: https://tools.ietf.org/html/rfc5789. Noudettu: 5.10.2019.

[16] Renzel D., Schlebusch P. & Klamma R. (2012) Today's top "restful" services and why they are not restful. In: International Conference on Web Information Systems Engineering, pp. 354–367. DOI: https://doi.org/10.1007/978-3-642-35063-4_26.

[17] Fielding R.T. (2008), On software architecture. Untangled. URL: https://roy.gbiv.com/untangled/2008/on-software-architecture. Noudettu: 19.10.2019.

[18] Fowler M. (2010), Richardson maturity model. martinfowler.com. URL: https://martinfowler.com/articles/richardsonMaturityModel.html. Noudettu: 7.10.2019.

[19] Wallace S. (2018), Why you should create an api definition and how to do its. SmartBear Software. URL: https://swagger.io/blog/api-development/why-you-should-create-an-api-definition/. Noudettu: 29.3.2020.

[20] Openapi specification 3.0.2 (2018). OpenAPI Initiative. URL: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.2.md. Noudettu: 30.9.2019.

[21] Myers G.J., Sandler C., Badgett T. & Thomas T.M. (2004) The art of software testing. John Wiley & Sons.

[22] Spillner A., Linz T. & Schaefer H. (2014) Software testing foundations: a study guide for the certified tester exam. Rocky Nook, Inc.

[23] Software fail watch: 5th edition. Tricentis. URL: https://www.tricentis.com/resources/software-fail-watch-5th-edition/. Noudettu: 28.3.2020.

[24] ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary (2017). International Organization for Standardization.

[25] Rafi D.M., Moses K.R.K., Petersen K. & Mäntylä M.V. (2012) Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In: 2012 7th International Workshop on Automation of Software Test (AST), pp. 36–42. DOI: https://doi.org/10.1109/IWAST.2012.6228988.

[26] Automated testing - understanding, designing, and setting up an effective automated testing strategy (2019). Postman, Inc. URL: https://www.postman.com/infographics/automated-testing-whitepaper.pdf. Noudettu: 26.3.2020.

[27] Fowler M. & Foemmel M. (2006), Continuous integration. martinfowler.com. URL: https://martinfowler.com/articles/continuousIntegration.html. Noudettu: 30.3.2020.

[28] Hilton M., Tunnell T., Huang K., Marinov D. & Dig D. (2016) Usage, costs, and benefits of continuous integration in open-source projects. In: 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 426–437.

[29] Tnt-fuzzer. URL: https://github.com/Teebytes/TnT-Fuzzer Noudettu: 15.1.2020.

[30] Apifuzzer (2018). URL: https://github.com/KissPeter/APIFuzzer Noudettu: 15.1.2020.

[31] Openapi testing meqanized. URL: https://github.com/meqaio/swagger_meqa Noudettu: 15.1.2020.

[32] Schemathesis. URL: https://github.com/kiwicom/schemathesis Noudettu: 5.1.2020.

[33] Got swag? URL: https://github.com/mobilcom-debitel/got-swag Noudettu: 15.1.2020.

[34] Openapi initiative faq (2019). OpenAPI Initiative. URL: https://www.openapis.org/faq. Noudettu: 25.9.2019.

[35] Smartbear assumes sponsorship of swagger api open source project (2015). SmartBear Software. URL: https://smartbear.com/news/news-releases/sponsorship-of-swagger. Noudettu: 27.9.2019.

[36] Openapi specification 3.0.0 (2017). OpenAPI Initiative. URL: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md. Noudettu: 10.1.2020.

[37] Openapi specification version 2.0 (2017). OpenAPI Initiative. URL: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md. Noudettu: 6.1.2020.

[38] Ed-Douibi H., Izquierdo J.L.C. & Cabot J. (2018) Automatic generation of test cases for rest apis: a specification-based approach. In: 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC), pp. 181–190. DOI: https://doi.org/10.1109/IWAST.2012.6228988.

[39] Martin-Lopez A., Segura S. & Ruiz-Cortés A. (2019) Test coverage criteria for restful web apis. In: Proceedings of the 10th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation, pp. 15–21. DOI: https://doi.org/10.1145/3340433.3342822.

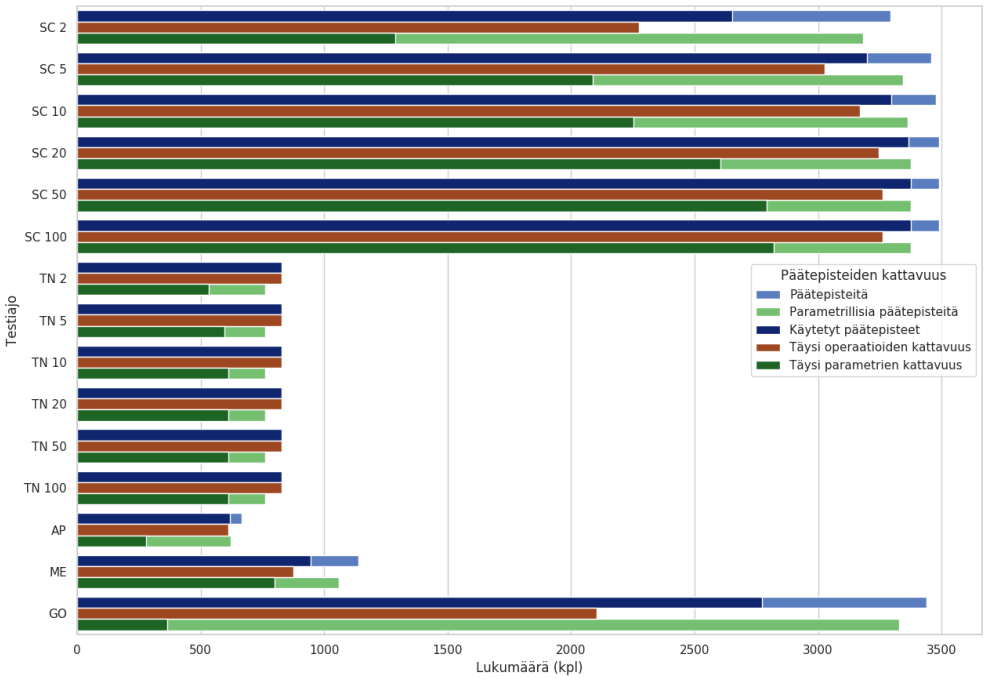
[40] Atlidakis V., Godefroid P. & Polishchuk M. (2018) Rest-ler: automatic intelligent rest api fuzzing .

9. LIITTEET

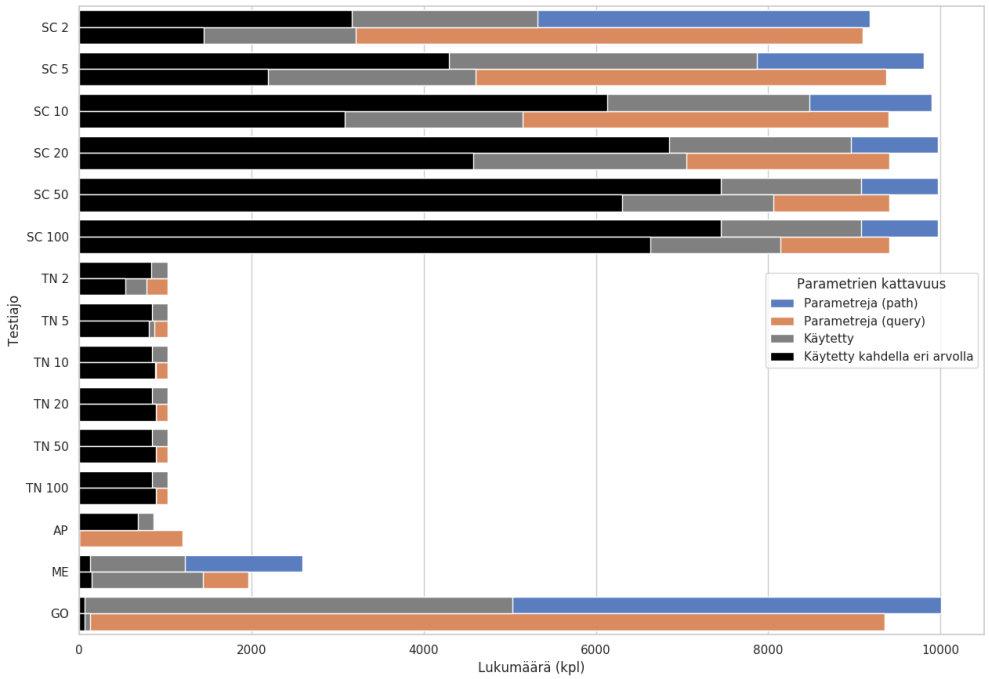
Liite 1 Testien tulokset
Liite 2 Creating developer tools by utilizing REST API Specification Coverage tool
Liite 3 Liite 3. Testin 1 testikohteet

Taulukko 6. Testi 1: Onnistuneiden testiajojen lukumäärät työkalu- ja suorituskohtaisesti

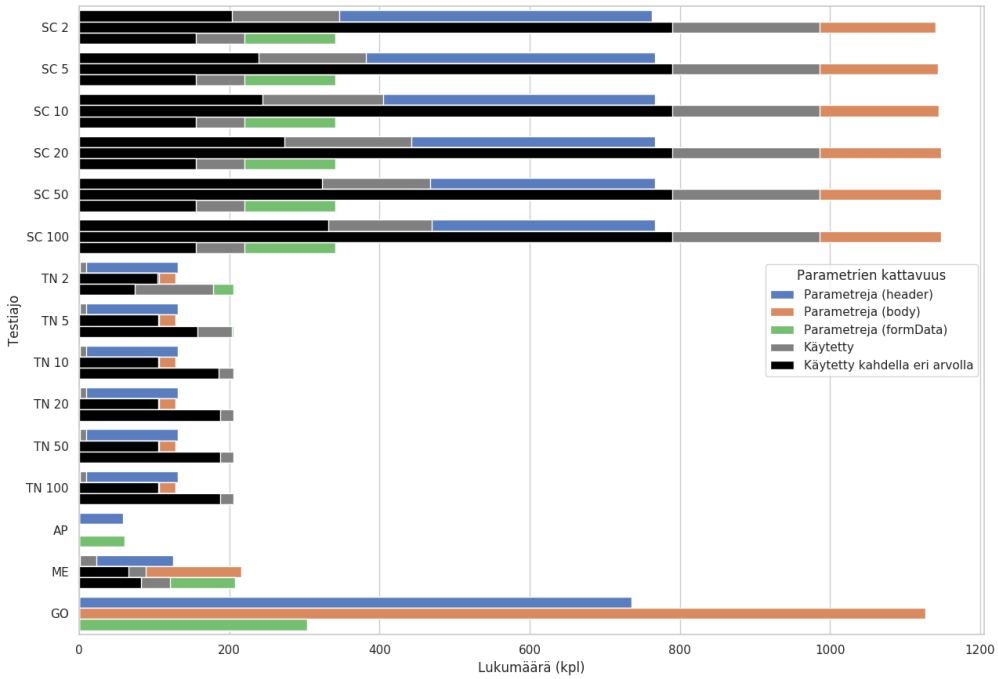
	Onnistuneet testiajot (kpl)	Onnistuneet testiajot (%)
SC 2	602	85,75
SC 5	677	96,44
SC 10	683	97,29
SC 20	689	98,15
SC 50	689	98,15
SC 100	689	98,15
TN 2	167	23,79
TN 5	167	23,79
TN 10	167	23,79
TN 20	167	23,79
TN 50	167	23,79
TN 100	167	23,79
AP	143	20,37
ME	242	34,47
GO	662	94,30



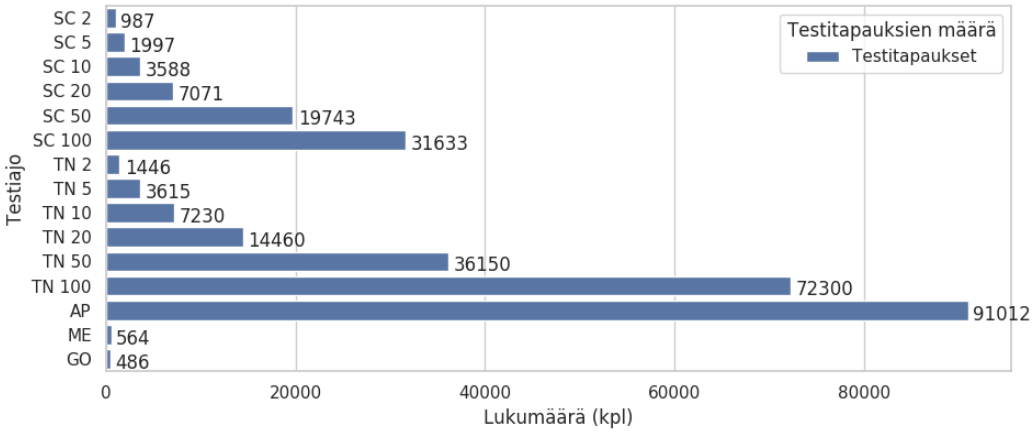
Kuva 16. Testi 1: Päätepisteiden kattavuus



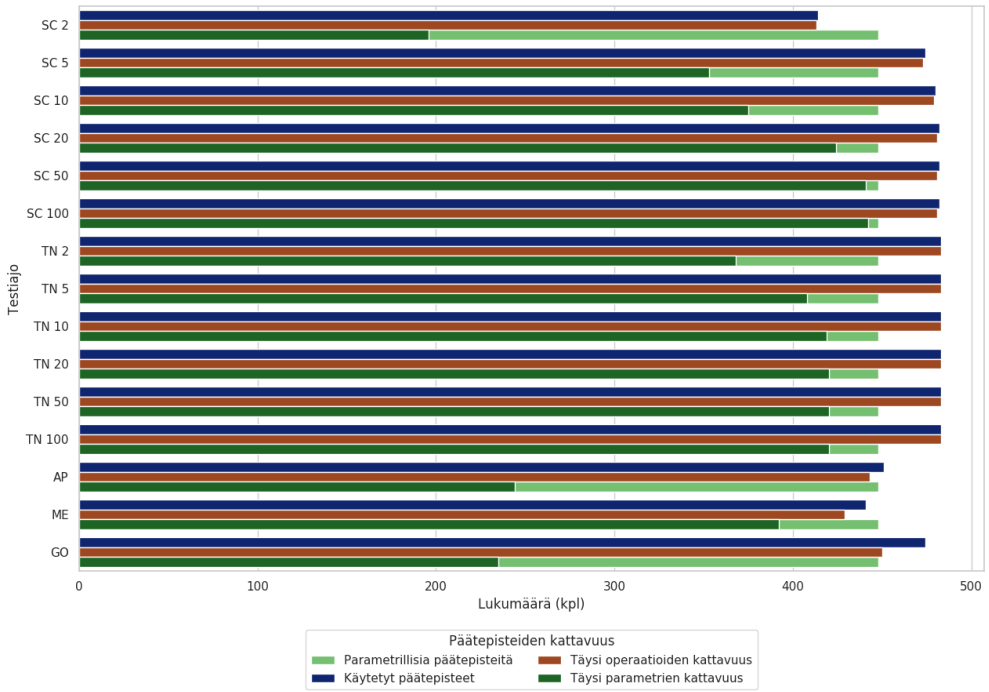
Kuva 17. Testi 1: Parametrien kattavuus, “path”- ja “query”-parametrit



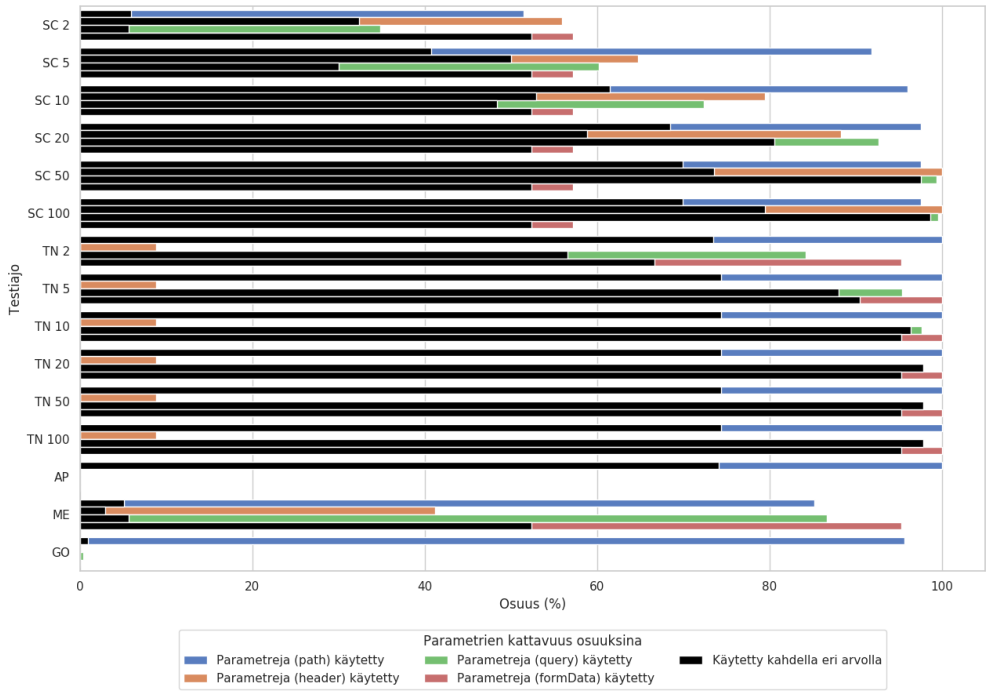
Kuva 18. Testi 1: Parametrien kattavuus, “header”-, “body” ja “formData”-parametrit



Kuva 19. Testi 1: Yksittäisten testitapausten yhteismäärä tarkasteltaessa kaikilla työkaluilla onnistuneesti testattuja rajapintoja



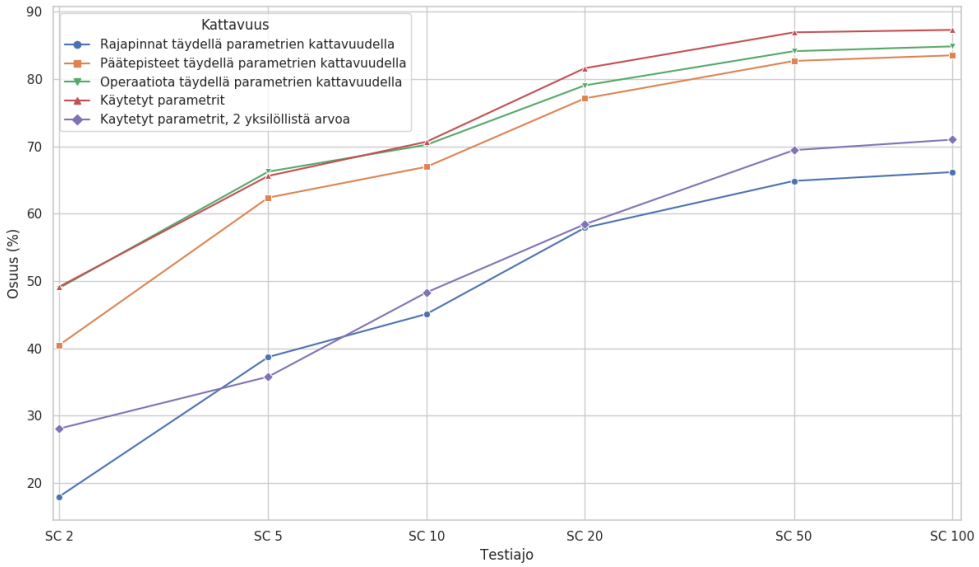
Kuva 20. Testi 1: Päätepisteiden kattavuus, kaikilla työkaluilla onnistuneesti testatut rajapinnat



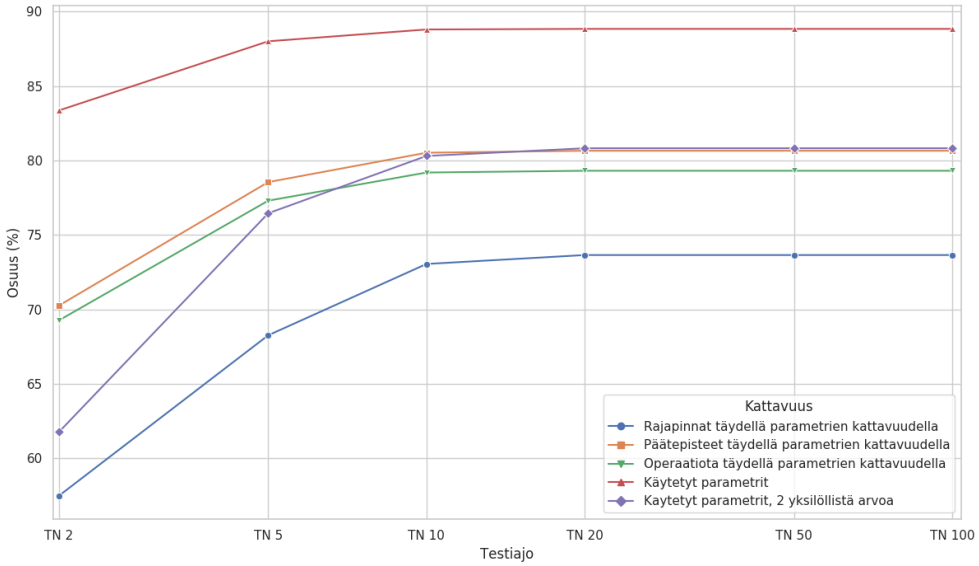
Kuva 21. Testi 1: Parametrien kattavuus osuuksina kaikilla työkaluilla onnistuneesti testatuilla rajapinnoilla, “path”-, “query”-, “header”- ja “formData”-parametrit

Taulukko 7. Testi 1: Työkalujen APIFuzzer, “Got Swag?” ja Meqa tuottamat suhteelliset parametrien kattavuudet

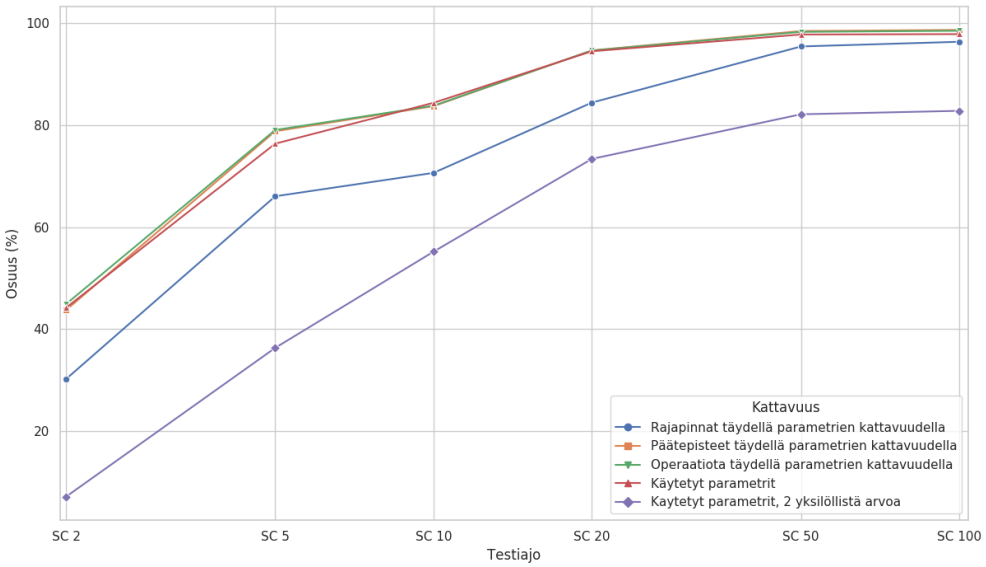
	AP	ME	GO
Käytetyt parametrit	39,40%	56,73%	23,97%
Käytetyt parametrit, 2 yksilöllistä arvoa	31,26%	8,28%	0,59%
Rajapinnat, täysi parametrien kattavuus	23,08%	46,69%	4,98%
Päätepisteet, täysi parametrien kattavuus	44,78%	75,38%	10,96%
Operaatiot, täysi parametrien kattavuus	44,33%	61,67%	9,52%



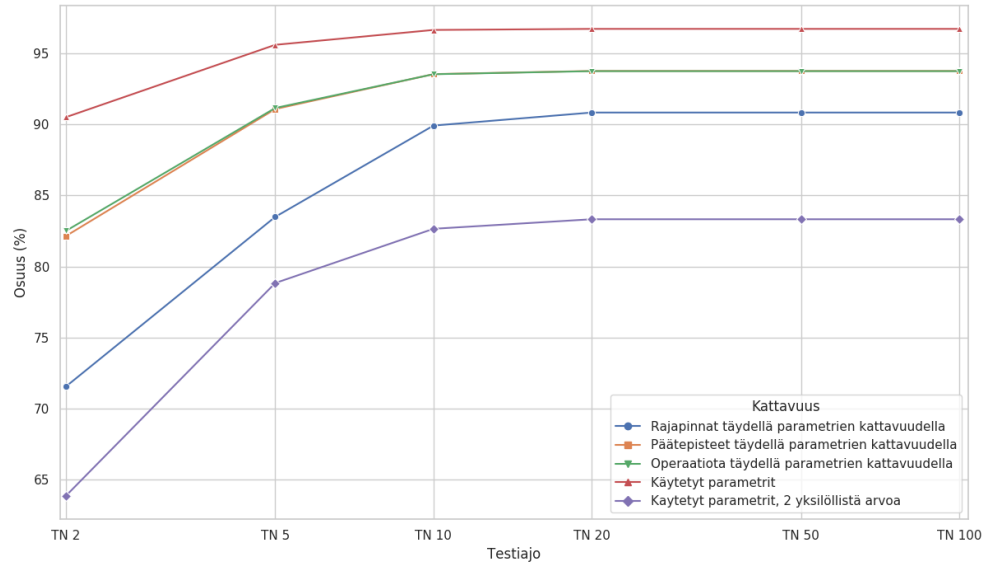
Kuva 22. Testi 1: Työkalun Schemathesis onnistuneesti testaamien rajapintojen sisältämien parametrien kattavuus suhteessa käytettyyn testitapausten luontimäärän asetukseen



Kuva 23. Testi 1: Työkalun TnT-Fuzzer onnistuneesti testaamien rajapintojen sisältämien parametrien kattavuus suhteessa käytettyyn testitapausten luontimäärän asetukseen



Kuva 24. Testi 1: Työkalun Schemathesis onnistuneesti testaamien rajapintojen sisältämien parametrien kattavuus suhteessa käytettyyn testitapausten luontimäärän asetukseen



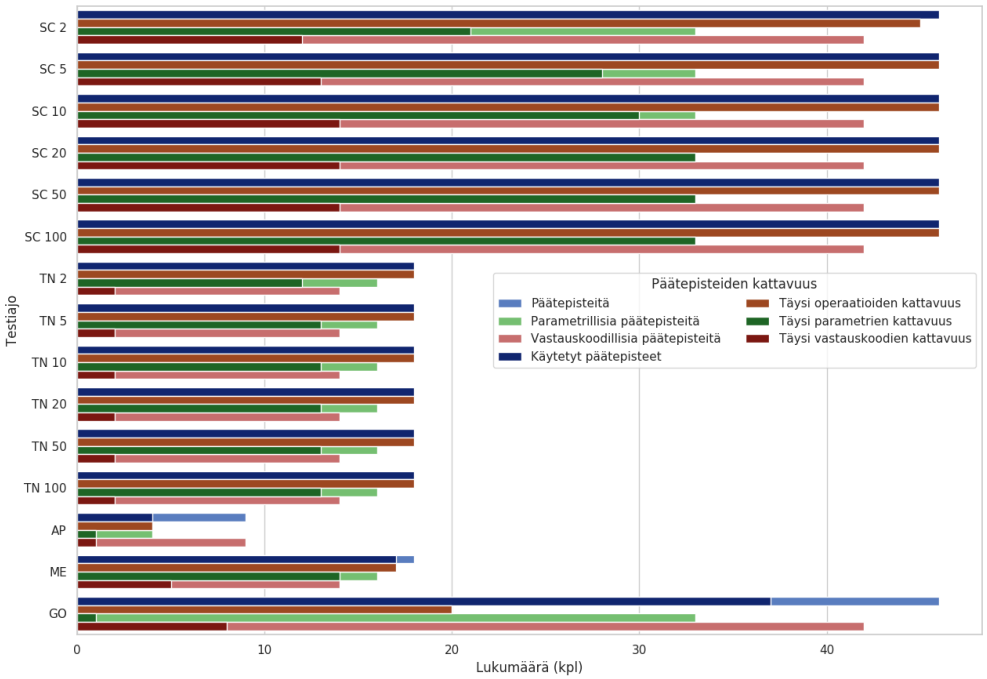
Kuva 25. Testi 1: Työkalun TnT-Fuzzer onnistuneesti testaamien rajapintojen sisältämien parametrien kattavuus suhteessa käytettyyn testitapausten luontimäärän asetukseen

Taulukko 8. Testi 1: Työkalujen APIFuzzer, “Got Swag?” ja Meqa tuottamat suhteelliset parametrien kattavuudet kaikilla työkaluilla onnistuneilla rajapinnoilla

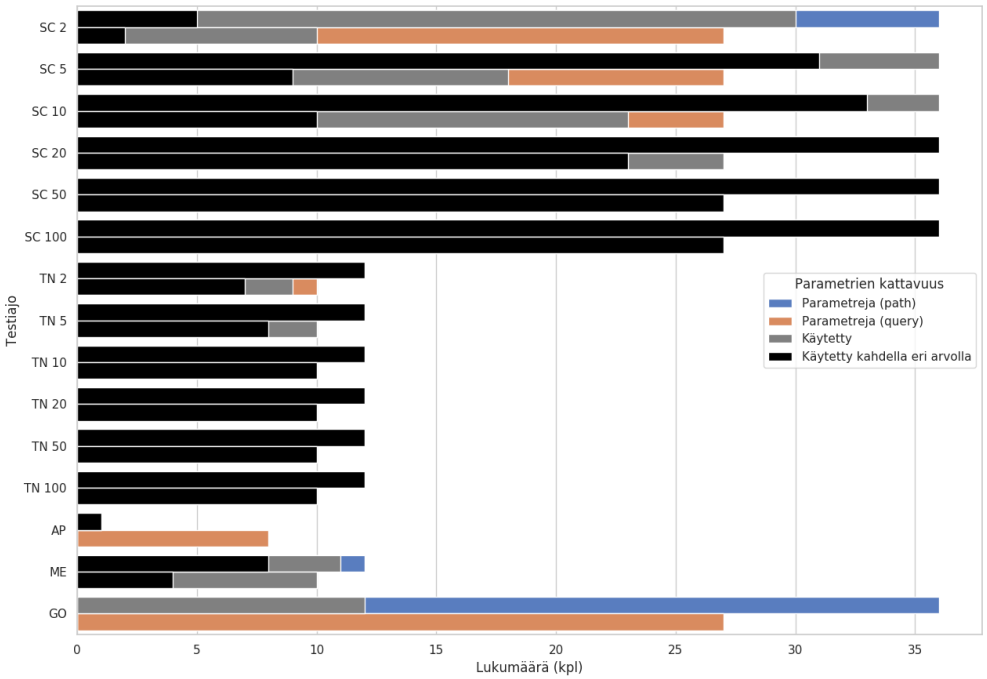
	AP	ME	GO
Käytetyt parametrit	51,01%	84,89%	48,92%
Käytetyt parametrit, 2 yksilöllistä arvoa	37,77%	6,06%	0,45%
Rajapinnat, täysi parametrien kattavuus	29,36%	72,48%	29,36%
Päätepisteet, täysi parametrien kattavuus	54,46%	87,50%	52,46%
Operaatiot, täysi parametrien kattavuus	54,86%	85,10%	52,92%

Taulukko 9. Testissä 2 käytettyjen rajapintojen määriteltyjen päätepisteiden, operaatioiden, vakiovastausten, vastauskoodien ja parametrien lukumäärät

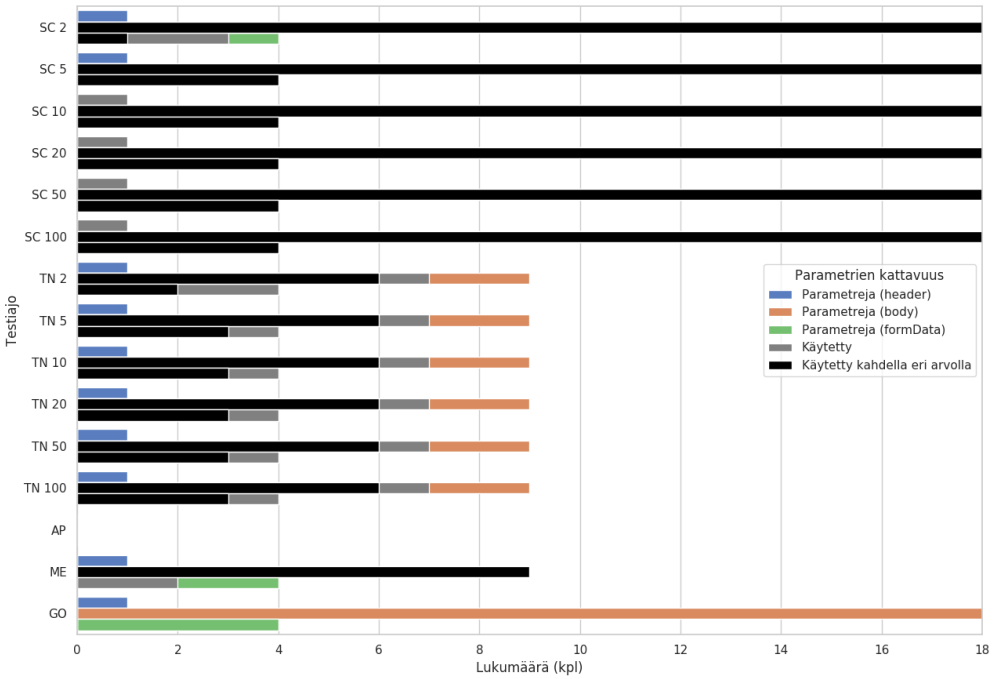
	Päätepisteet	Operaatiot	Vakiovastaukset	Vastauskoodit	Parametrit
Swagger Petstore	14	20	4	32	25
VideoGameDB	2	5	0	5	5
Mailhog	2	2	0	2	6
Prometheus Alertmanager	6	9	0	21	16
ORY Oathkeeper	7	7	0	17	3
Jupyter Notebookserver	15	29	0	54	31



Kuva 26. Testi2: Päätepisteiden kattavuus



Kuva 27. Testi 2: Parametrien kattavuus, “path”- ja “query”-parametrit



Kuva 28. Testi 2: Parametrien kattavuus, “header”-, “body”- ja “formData”-parametrit

Creating developer tools by utilizing REST API Specification Coverage tool

Final Report

Name	Student number
Nikunj Arora	(Redacted)
Joonas Hilke	(Redacted)
Santeri Moberg	(Redacted)
Antti Männikkö	(Redacted)
Juuso Säärelä	(Redacted)

Executive abstract

This report was made as a part of University of Oulu's course "Software Project". In this report, the solution to customer's problem, easier integration to a development pipeline and need for better looking output for their own product, is examined and presented.

The report presents customer's current product, the problem, motivation, ideas to solve the problem and the implemented software solution. It goes in deep detail into how the product was made, what is the final architecture and how agile methods were utilized.

Customer's wish was creating better human readable output as well as easier way to use their own program - API Specification Coverage Tool (ASC). The solution comes in to parts: the HTTP/s man in the middle-proxy to capture HAR files for OpenAPI testing and GUI part to make the information easily readable.

Project Description and Realised Requirements chapters go into detail why the proxy implementation was chosen, how it was planned and making of the solution. The next two chapters depict the software development methods and testing that was done. Last chapter gives feedback to the course organizers.

Table of Contents

1. Introduction

Software testing is an integral part of software development; many development tools come with debugging tools or their own testing scripts, development teams measure their software quality through code coverage, and so forth. New technology for finding holes and testing the source code is constantly being made.

Many test tools are fairly straightforward: run script (usually through command line) and display information on screen or generated report. However, some tools seem very primitive on what they do or do not display on the test information. While certain test tools seem to be great at reducing the time to find problems in code and assure software quality, the amount of time to set up and get the tests running makes them frustrating to use. Also, the output of the test tool might not be ideal for the developer and looking through the results might prove to be a lot of work.

This course report presents solution for customer's problem, where they have working REST API specification coverage tool called ASC tool [1]. The main problem of integrating usage of ASC tool to CI/CD pipeline is that it requires HAR file of the test traffic as an input and generating a HAR file requires manual work from the user. Project provides a solution for generating HAR files through the use of proxy and thus making usage of the ASC easier. A simple GUI was also developed for the output of the ASC to make the job of finding the essential information from the output easier for the user.

2. Customer

Customers of the project were Jukka Pajukangas and Pekka Pietikäinen from University of Oulu.

3. Acronyms and Definitions

API	Application program interface
ASC	Api specification coverage
CI/CD	Continuous Integration and Delivery
GUI	Graphical User Interface
HAR	HTTP archive file format
HTTP	Hypertext transfer protocol
OpenAPI	API description format for REST APIs
REST	Representational State Transfer
TCP	Transmission Control Protocol

4. References

[1] API Specification Coverage Tool. Available: https://github.com/ouspg/ASC (Accessed November 21, 2019)

[2] mitmproxy. Available: https://mitmproxy.org/(Accessed December 16, 2019)

[3] Postman. Available: https://www.getpostman.com/ (Accessed December 30, 2019)

[4] PySide2. Available: https://pypi.org/project/PySide2/ (Accessed December 29, 2019)

[5] Sutherland, J., & Schwaber, K. (2013). The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game. Scrum. org.

[6] Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). Manifesto for agile software development.

5. Project Description

The project's motive is to make the API Specification coverage tool (ASC) [1] easier to use so that it can be integrated in development pipeline. On its own, the ASC requires a lot of setting up and manual work to run the software which makes it hard for API developer to use. The output of the program is not so easy to read, making it harder to utilize during API testing.

The ASC [1] is a python program that is developed by our customer, meant for showing REST API coverage as well as anomalies. The inputs it uses are the OpenAPI specification file of the API that is being tested, HAR-file of the test traffic and a configuration file that contains the server address and basepath of the API among other settings. The basepath and server address can be also overridden with command line arguments when running the ASC. After ASC has ran, it generates a report that contains all the endpoints that were listed in the OpenAPI specification with their coverage as well as anomalies, e.g. a response code that was not defined in the OpenAPI specification, that were encountered. The tool is still in development.

Many different ideas ranging from just simple GUI to bigger applications were had. One of the first promising ideas was to create a packet sniffer that would capture web traffic data. This was quickly abandoned because of the disadvantages of that method. The reassembling of a pcap file to a file containing TCP traffic which then would be formed to a HAR file would have been troublesome in multiple ways, e.g. how would missing packets be handled and how the TCP file would be formed and structured. HTTPS content would have also been very difficult to analyze with a sniffer due to encryption. Two other good sounding ideas were an IDE plugin and HTTP proxy for capturing the HAR file. After some discussion on the specifications and usable technology, a slightly modified idea of the HTTP proxy was considered doable.

The application comes in two parts: the GUI client for the output and a proxy listener. The motivation for this kind of architecture is to make using the application as easy as possible to use as well as develop and maintain. After inputting server address, OpenAPI specification and basepath, the application creates a HTTP proxy that is able to capture the the test traffic. After the tests have been ran, user closes the proxy which then generates a HAR-file that is used for running the ASC. Once the ASC has ran, its output is displayed to the user. User can decide if they want an output with GUI or a print in the command line interface. The source code is hosted on Gitlab[1].

5.1 Deployment

Diagram of the deployed system is presented in figure 1. Deployed system consists of several subsystems which communicate with each other: Application, proxy, GUI, API Server and ASC. When running the application user inputs the OpenAPI specification of the api, server address of the api and basepath of the api. Application then generates a HTTP proxy that is able to capture the test traffic to the api and save it as a HAR file. After user

[1] https://gitlab.com/juusis/softwareproject2019

has ran the tests, they close the proxy through command line interface and then the application forwards the HAR file and OpenAPI specification file of the api to the ASC which output is then displayed with a GUI client.

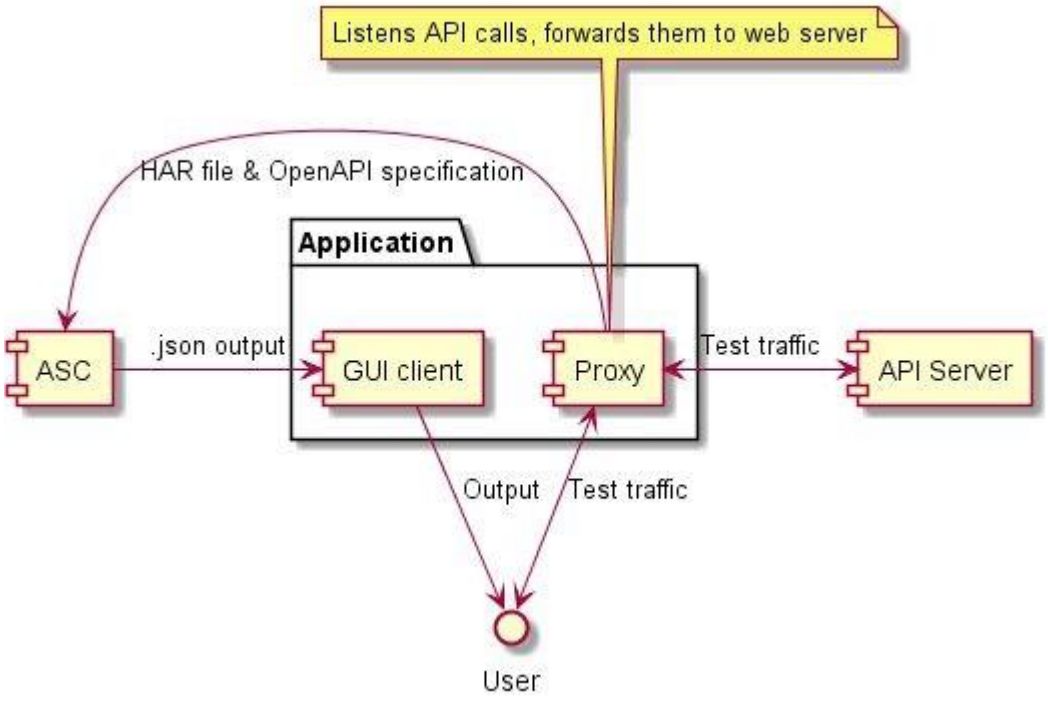


Figure 1: Diagram of the deployed system.

6. Realised requirements

Since ASC tool is already robust itself, there is no need to make modifications at the source code. Rather, the software that was developed is more of an wrapper around it which helps to make usage of ASC more convenient for OpenAPI developers.

When the decision to make a proxy listener was made, the first object was to find the right technology. Mitmproxy was selected for it containing all the needed features for our project: being a HTTP proxy that can forward requests as well as having a Python API [2]. Using mitmproxy, the application creates a reverse proxy to localhost:8080 address and is then able to save the traffic that goes through it as a HAR file when user closes the reverse proxy.

Setting a proxy to the address that the reverse proxy is listening differs between Windows and Unix versions of the software. In Windows version, this is done automatically by the application where it generates system wide proxy by changing registry keys. In the Unix version, making a proxy for the test traffic is handled by the user. This can be easily done inside an application such as Postman [3] or by exporting http proxy inside terminal which is meant to run the tests to the api.

6.1 The structure

There are GUI and GUI-less Windows and Unix versions of the program. The GUI-less versions only print the ASC output to the command line interface.

If user has chosen the version with GUI script, gui.py is called. For creating the GUI, PySide2 (Python module for Qt) is used [4]. The GUI script uses the json report generated by the ASC tool to dynamically generate color coded buttons for different endpoints. The logic for the color coding is presented in figure 2.

```
68      def CreateButtons(self):
69          """ Creates method buttons dynamically based on data. """
70
71          buttons = []
72          for i, j in self.methods.items():
73              button = QPushButton(i)
74              button.clicked.connect(self.showData)
75              if (j["response_codes_count"] == j["response_codes_used"] and
76                  j["parameters_used"] == j["parameters_count"] and
77                  j["default_response_exists"] == False and j["anomalies_count"] == 0):
78                  button.setStyleSheet(self.stylesheetButtonGreen)
79              elif j["response_codes_used"] == 0 and j["parameters_used"] == 0:
80                  button.setStyleSheet(self.stylesheetButton)
81              else:
82                  button.setStyleSheet(self.stylesheetButtonRed)
83
84              self.layout.addWidget(button)
```

Figure 2: Logic of the colour coding.

Different screens of the GUI are presented in figures 3-5. In figure 3, the main screen of the GUI shows the fully tested endpoints that contain no anomalies as green, the endpoints that were not fully tested or contained anomalies as red, and if an endpoint was not covered in testing the GUI displays that endpoint as grey.

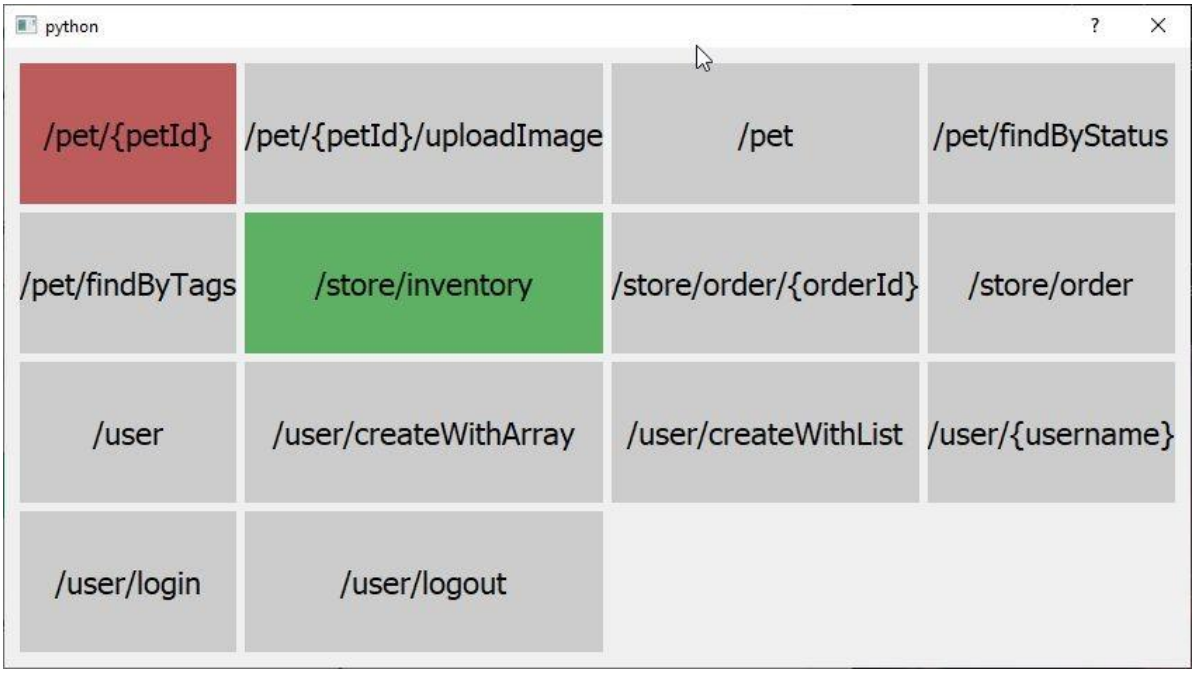


Figure 3: GUI of the output.

In figure 4, pop up window of specific endpoint is presented. The submenu is opened by clicking an endpoint from the main menu. The submenu presents each HTTP method of a clicked endpoint. The color coding in this

submenu is the same as in the previous menu and is meant to help pinpoint the issues in coverage or anomalies to the user.

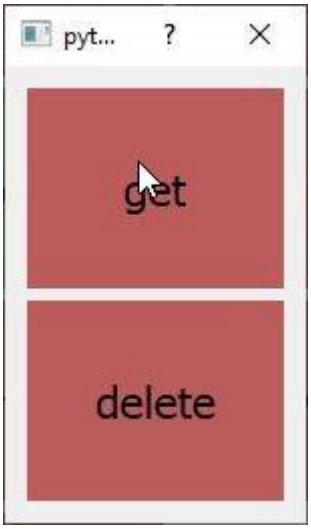


Figure 4: Submenu of one of the endpoint buttons.

When clicking a method button in the submenu from figure 4, a json print containing all the information about that specific method is presented to the user. The information can be seen from figure 5.

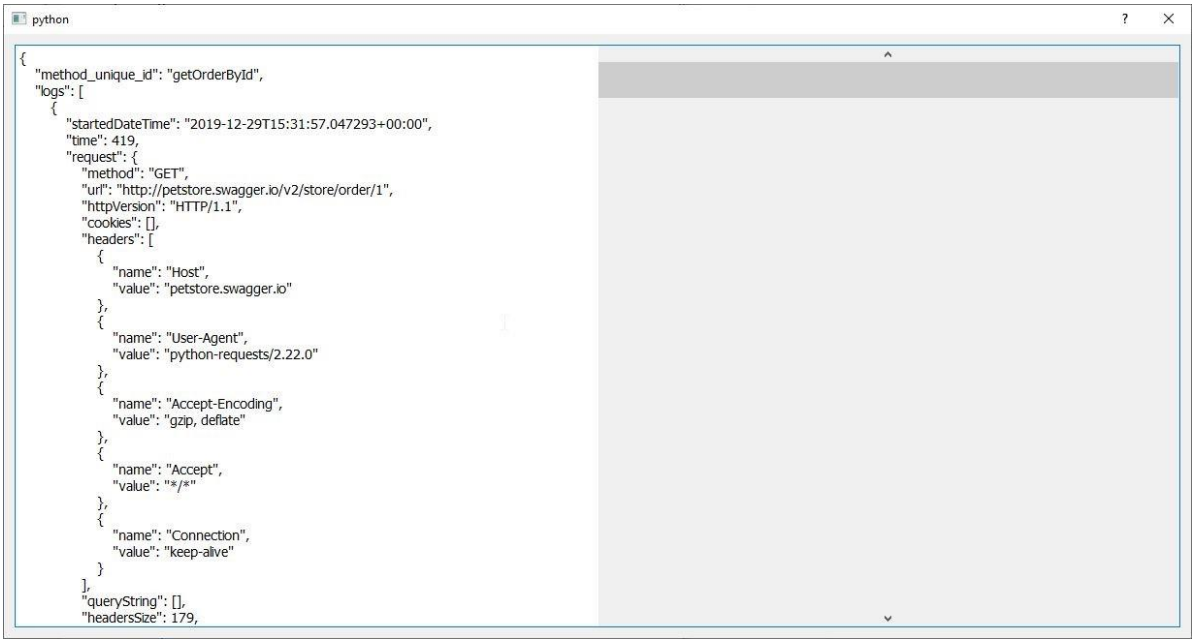


Figure 5: Information about a specific method in json format.

6.2 Issues and future work

In the Windows version of the software, there is a bug which leads to system wide proxy sometimes not starting. It can fixed by opening Network & Internet settings menu on windows 10. The bug is a major problem in the usability of the windows version and fixing it is a big priority in the future if the development continues. The suspected main cause for this bug is that while we do change the registry key values they need to be read from

the registry by some application before they take effect. By opening the the settings, values are read and system wide proxy becomes active.

The GUI at its present state does not clearly show the anomalies found by the ASC tool. Anomalies found can be seen from the final submenu's json print as a list or from anomaly_report.txt generated by the ASC tool. The anomaly report could be integrated to the GUI in a similar fashion as the buttons for the endpoints and methods in the future.

7. Used software development method

The course suggested teams to utilize Scrum. While the team initially followed the suggested method, in the end the team's method was closer to Kanban:

- No Scrum Master and (Internal) Product Owner. Everyone took care of Trello board and product backlog.
- No daily stand-up
- While there were bi-weekly reviews, development was more of ad-hoc than in sprints.
- No sprint retros or goals.
- Trello board was used as workflow management tool

While Scrum is easy to understand, implementing it is quite demanding. Schwaber and Sutherland, the forerunners of Scrum, state in their Scrum guide that " ...although implementing only parts of Scrum is possible, the result is not Scrum" [5].

The second focus of the course was agile development and testing. While the team did not follow the Scrum guidelines, the teams workflow could be described agile. The distinguishing features of any agile method - iterative development, frequent physical meetings and customer collaboration - were had [6].

8. Testing / acceptance of the software

Unit testing was done using Python's unittest library. It is fairly standard testing library with assertions. Tests can be run by running tests.py file.

The proxy is integral part of the program. Assuring that the proxy starts and resets was main concern; because of that, unit tests focus more on proxy instead of GUI part. Tests to make sure that the GUI correctly renders and buttons work were made with unittest and PySide2's own testing module.

In the development, for the simulated test data we used example api called petstore[2] that is hosted by Swagger. We used a python library called Schemathesis[3] to automatically generate test cases based on the OpenAPI specification of the petstore.

At the end of (almost every) sprint was demo where customer reviewed the progress and stated what they thought. Though there was no evaluation on source code side during these demos, the customer's feedback during these were important.

At December 20th final product review was had. Customer checked that the specifications were met and gave feedback on overall product. Some small fixes were suggested before returning the project work but overall client was satisfied.

[2] http://petstore.swagger.io/
[3] https://github.com/kiwicom/schemathesis

9. Project timeline

September 4th, 2019: The course starts.

September 6th, 2019: The project presentation, choosing the project.

September 9th, 2019: First real meeting with customer, some thinking about the solution. The first Sprint starts.

September 9-20th: Discussion and experimentation with different technologies. Familiarization with ASC.

September 20th: First Sprint demo. Discussion on ideas, what could work what could not. Choosing few concepts that look promising and further investigation.

September 23-October 4th: Second sprint. More research and discussion on project structure, some early prototyping. Testing with sniffer and HTTP/S proxy.

October 4th: Second Sprint demo. Deciding on focusing on HTTP/S proxy.

October 7-18th: Experimenting with sniffer and proxy. Some more architecture prototyping. Test traffic making. First tests with mitmproxy.

October 18th: Planning around the proxy architecture. Discussion on what parts need to be implemented.

October 21-24th: Some small discussion. mitmproxy saves HAR. Making preparations for midterm presentation.

October 25th: Midterm presentation.

October 28-November 1st: Focus on other projects. Only small discussion.

November 4-15th: Developing integration between mitmproxy and ASC tool. Writing of the documentation begins.

November 15th: Sprint Demo. Some discussion about what works and what needs improvement.

November 18-29th: Quieter time between development. Some small changes such as refactoring.

December 2-14th: Finalization of proxy part. GUI planning and development starts.

December 16-20th: GUI done, writing unit tests.

December 20th: Final meeting with customer. Some small feedback.

December 29-30th: Final report writing and submission.

10. Feedback to the course organizers

We would have wished a little more interaction on the course organizers part. The beginning of the course felt a bit rushed, as we were supposed to pick a team and a project in a few days. There was only midterm presentation between the starting lecture and deadline - even a small status presentation to the organizers every start of the month would have made the course more pleasant. Some autonomy is good - project team was lucky that the product owner was aboard the development process through constant feedback and product review. However, some guidance and knowledge or regular feedback of the course organizers would perhaps made the solution even better.

There were some argumentation whether Scrum fits to the course. As stated earlier, it is fairly easy to describe it, but it also needs serious dedication to the development. Processes such as daily stand-ups and retro are not feasible in the scope of the course - they need more devoted teams and higher stakes. Methods such as Lean

Software Development or Kanban would probably be more sufficient, while also teaching students about what is agile development in practise.

Taulukko 10. Testi 1: Kaikilla työkaluilla onnistuneesti testatut rajapinnat

Otsikko	Palvelun nimi	Palveluntarjoaja
Afterbanks		afterbanks.com
airportsapi		airport-web.appspot.com
AviationData.Systems Airports API V1		aviationdata.systems
ApiManagementClient	apimanagement-apimapisByTags	azure.com
ApiManagementClient	apimanagement-apimissues	azure.com
ApiManagementClient	apimanagement-apimnetworkstatus	azure.com
ApiManagementClient	apimanagement-apimpolicysnippets	azure.com
ApiManagementClient	apimanagement-apimproductsByTags	azure.com
ApiManagementClient	apimanagement-apimregions	azure.com
ApiManagementClient	apimanagement-apimtagresources	azure.com
ApplicationInsightsManagementClient	applicationinsights-aiOperations_API	azure.com
WorkbookClient	applicationinsights-workbookOperations_API	azure.com
AutomationManagement	automation-linkedWorkspace	azure.com
Update Management	automation-softwareUpdateConfigurationMachineRun	azure.com
Update Management	automation-softwareUpdateConfigurationRun	azure.com
AutomationManagement	automation-sourceControlSyncJobStreams	azure.com
StorageManagementClient	azsadmin-acquisitions	azure.com
StorageManagementClient	azsadmin-blobServices	azure.com
SubscriptionsManagementClient	azsadmin-DelegatedProvider	azure.com
SubscriptionsManagementClient	azsadmin-DelegatedProviderOffer	azure.com
ComputeDiskAdminManagementClient	azsadmin-Disks	azure.com
AzureBridgeAdminClient	azsadmin-DownloadedProduct	azure.com
GalleryManagementClient	azsadmin-Gallery	azure.com
AzureBridgeAdminClient	azsadmin-Product	azure.com
StorageManagementClient	azsadmin-queueServices	azure.com
StorageManagementClient	azsadmin-shares	azure.com
StorageManagementClient	azsadmin-tableServices	azure.com
NetworkManagementClient	network-availableDelegations	azure.com
NetworkManagementClient	network-azureFirewallFqdnTag	azure.com
NetworkManagementClient	network-checkDnsAvailability	azure.com
NetworkManagementClient	network-endpointService	azure.com
NetworkManagementClient	network-operation	azure.com
NetworkManagementClient	network-serviceCommunity	azure.com
NetworkManagementClient	network-serviceTags	azure.com
NetworkManagementClient	network-usage	azure.com
Security Center	security-operations	azure.com
Security Center	security-serverVulnerabilityAssessments	azure.com
Bandsintown		bandsintown.com
Cenit IO - REST API Specification		cenit.io
Cnab Online		cnab-online.herokuapp.com
The Consumer Financial Protection Bureau		consumerfinance.gov
Betriebsstellen	betriebsstellen	deutschebahn.com
Fahrplan-Free	fahrplan	deutschebahn.com
FaSta-Station_Facilities_Status	fasta	deutschebahn.com
Flinkster_API_NG	flinkster	deutschebahn.com
Reisezentren-API	reisezentren	deutschebahn.com
Stationsdatenbereitstellung	stada	deutschebahn.com
domainsdb.info		domainsdb.info
Item Feed Service	buy-feed	ebay.com
Buy Marketing	buy-marketing	ebay.com
Faretrotter Travel		faretrotter.com
Freesound		freesound.org
Furkot Trips		furkot.com
Highways England API		highwaysengland.co.uk
HSBC UK		hsbc.com
import.io	data	import.io
import.io	extraction	import.io
import.io	rss	import.io
Instawell		instawell.com
ISBNdb		isbndb.com
LetMC Api V3, reporting	reporting	letmc.com
Musixmatch		musixmatch.com
Native Ads Publisher		nativeads.com
nFusion Solutions Market Data API v1		nfusionsolutions.biz
Archive	archive	nytimes.com
Article Search	article_search	nytimes.com
Semantic	semantic_api	nytimes.com
Times Newswire	timeswire	nytimes.com
Top Stories	top_stories	nytimes.com
Mobility	mobility	o2.cz
Socio-demo	sociodemo	o2.cz
Swagger2OpenAPI Converter		openapi-converter.herokuapp.com
ODN		opendatanetwork.com
Polling Places	pollingplaces	phila.gov
Polygon		polygon.io
Quicksold	location	quicksold.co.uk
Car Registration		regcheck.org.uk
Rotten Tomatoes		rottentomatoes.com
SchoolDigger API V1		schooldigger.com
Sonar Trading		sonar.trading
CBB v3 Scores	cbb-v3-scores	sportsdata.io
CFB v3 Scores	cfb-v3-scores	sportsdata.io
CS:GO v3 Scores	csgo-v3-scores	sportsdata.io
CS:GO v3 Stats	csgo-v3-stats	sportsdata.io
Golf v2	golf-v2	sportsdata.io
LoL v3 Projections	lol-v3-projections	sportsdata.io
LoL v3 Scores	lol-v3-scores	sportsdata.io
MLB v3 Play-by-Play	mlb-v3-play-by-play	sportsdata.io
MLB v3 Projections	mlb-v3-projections	sportsdata.io
MLB v3 RotoBaller Articles	mlb-v3-rotoballer-articles	sportsdata.io
MLB v3 RotoBaller Premium News	mlb-v3-rotoballer-premium-news	sportsdata.io
MLB v3 Scores	mlb-v3-scores	sportsdata.io

NASCAR v2	nascar-v2	sportsdata.io
NBA v3 Play-by-Play	nba-v3-play-by-play	sportsdata.io
NBA v3 Projections	nba-v3-projections	sportsdata.io
NBA v3 RotoBaller Articles	nba-v3-rotoballer-articles	sportsdata.io
NBA v3 RotoBaller Premium News	nba-v3-rotoballer-premium-news	sportsdata.io
NFL v3 Play-by-Play	nfl-v3-play-by-play	sportsdata.io
NFL v3 Projections	nfl-v3-projections	sportsdata.io
NFL v3 RotoBaller Articles	nfl-v3-rotoballer-articles	sportsdata.io
NFL v3 RotoBaller Premium News	nfl-v3-rotoballer-premium-news	sportsdata.io
NHL v3 Play-by-Play	nhl-v3-play-by-play	sportsdata.io
NHL v3 Projections	nhl-v3-projections	sportsdata.io
Soccer v3 Projections	soccer-v3-projections	sportsdata.io
Storm Glass Marine Weather		stormglass.io
Locations	locations	whapi.com
Numbers	numbers	whapi.com
Winning Email		winning.email
XKCD		xkcd.com

Taulukko 11. Testi 1: Rajapinnat, joita ei testattu onnistuneesti kaikilla työkaluilla

Otsikko	Palvelun nimi	Palveluntarjoaja
Authentiq		6-dot-authentiqio.appspot.com
AIception Interactive		aiception.com
Amazon Cognito Sync	cognito-sync	amazonaws.com
AWS IoT Data Plane	iot-data	amazonaws.com
Qakka	qakka	apache.org
Search Services	search	archive.org
Wayback	wayback	archive.org
Authentiq Connect		authentiq.io
Axesso Api		axesso.de
Azure Addons Resource Provider	addons-Addons	azure.com
Azure Addons Resource Provider	addons-addons-swagger	azure.com
AdvisorManagementClient	advisor	azure.com
Azure Alerts Management Service Resource Provider	alertsmanagement-SmartDetectorAlertRulesApi	azure.com
AzureAnalysisServices	analysisservices	azure.com
ApiManagementClient	apimanagement-apimapiversionsets	azure.com
ApiManagementClient	apimanagement-apimauthorizationservers	azure.com
ApiManagementClient	apimanagement-apimbackends	azure.com
ApiManagementClient	apimanagement-apimcaches	azure.com
ApiManagementClient	apimanagement-apimcertificates	azure.com
ApiManagementClient	apimanagement-apimdeployment	azure.com
ApiManagementClient	apimanagement-apimdiagnostics	azure.com
ApiManagementClient	apimanagement-apimemailtemplate	azure.com
ApiManagementClient	apimanagement-apimemailtemplates	azure.com
ApiManagementClient	apimanagement-apimgroups	azure.com
ApiManagementClient	apimanagement-apimidentityprovider	azure.com
ApiManagementClient	apimanagement-apimloggers	azure.com
ApiManagementClient	apimanagement-apimnotifications	azure.com
ApiManagementClient	apimanagement-apimopenidconnectproviders	azure.com
ApiManagementClient	apimanagement-apimpolicies	azure.com
ApiManagementClient	apimanagement-apimproducts	azure.com
ApiManagementClient	apimanagement-apimproperties	azure.com
ApiManagementClient	apimanagement-apimquotas	azure.com
ApiManagementClient	apimanagement-apimreports	azure.com
ApiManagementClient	apimanagement-apimsubscriptions	azure.com
ApiManagementClient	apimanagement-apimtags	azure.com
ApiManagementClient	apimanagement-apimtenant	azure.com
ApiManagementClient	apimanagement-apimusers	azure.com
ApiManagementClient	apimanagement-apimversionsets	azure.com
ApplicationInsightsManagementClient	applicationinsights-analyticsItems_API	azure.com
ApplicationInsightsManagementClient	applicationinsights-componentAnnotations_API	azure.com
ApplicationInsightsManagementClient	applicationinsights-componentApiKeys_API	azure.com
ApplicationInsightsManagementClient	applicationinsights-componentContinuousExport_API	azure.com
ApplicationInsightsManagementClient	applicationinsights-componentFeaturesAndPricin...	azure.com
ApplicationInsightsManagementClient	applicationinsights-componentProactiveDetectio...	azure.com
ApplicationInsightsManagementClient	applicationinsights-components_API	azure.com
ApplicationInsightsManagementClient	applicationinsights-componentWorkItemConfigs_API	azure.com
ApplicationInsightsManagementClient	applicationinsights-favorites_API	azure.com
Azure Log Analytics Query Packs	applicationinsights-QueryPackQueries_API	azure.com
Azure Log Analytics Query Packs	applicationinsights-QueryPacks_API	azure.com
ApplicationInsightsManagementClient	applicationinsights-webTestLocations_API	azure.com
ApplicationInsightsManagementClient	applicationinsights-webTests_API	azure.com
ApplicationInsightsManagementClient	applicationinsights-workbooks_API	azure.com
AttestationManagementClient	attestation	azure.com
AuthorizationManagementClient	authorization	azure.com
AuthorizationManagementClient	authorization-authorization-ClassicAdminCalls	azure.com
AuthorizationManagementClient	authorization-authorization-DenyAssignmentGetC...	azure.com
AuthorizationManagementClient	authorization-authorization-ProviderOperations...	azure.com
AuthorizationManagementClient	authorization-authorization-RACalls	azure.com
AuthorizationManagementClient	authorization-authorization-RoleAssignmentsCalls	azure.com
AuthorizationManagementClient	authorization-authorization-RoleBasedCalls	azure.com
AuthorizationManagementClient	authorization-authorization-RoleDefinitionsCalls	azure.com
AutomationManagement	automation-account	azure.com
AutomationManagement	automation-certificate	azure.com
AutomationManagement	automation-connection	azure.com
AutomationManagement	automation-connectionType	azure.com
AutomationManagement	automation-credential	azure.com
AutomationManagement	automation-dscCompilationJob	azure.com
AutomationManagement	automation-dscConfiguration	azure.com
AutomationManagement	automation-dscNode	azure.com
AutomationManagement	automation-dscNodeConfiguration	azure.com
AutomationManagement	automation-dscNodeCounts	azure.com
AutomationManagement	automation-hybridRunbookWorkerGroup	azure.com

AutomationManagement	automation-job	azure.com
AutomationManagement	automation-jobSchedule	azure.com
AutomationManagement	automation-module	azure.com
AutomationManagement	automation-python2package	azure.com
AutomationManagement	automation-runbook	azure.com
AutomationManagement	automation-schedule	azure.com
Update Management	automation-softwareUpdateConfiguration	azure.com
AutomationManagement	automation-sourceControl	azure.com
AutomationManagement	automation-sourceControlSyncJob	azure.com
AutomationManagement	automation-variable	azure.com
AutomationManagement	automation-watcher	azure.com
AutomationManagementClient	automation-webhook	azure.com
SubscriptionsManagementClient	azsadmin-AcquiredPlan	azure.com
AzureBridgeAdminClient	azsadmin-Activation	azure.com
InfrastructureInsightsManagementClient	azsadmin-Alert	azure.com
FabricAdminClient	azsadmin-ApplicationOperationResults	azure.com
AzureBridgeAdminClient	azsadmin-AzureBridge	azure.com
BackupManagementClient	azsadmin-Backup	azure.com
BackupManagementClient	azsadmin-BackupLocations	azure.com
BackupManagementClient	azsadmin-Backups	azure.com
CommerceManagementClient	azsadmin-Commerce	azure.com
CommerceManagementClient	azsadmin-CommerceAdmin	azure.com
Compute Admin Client	azsadmin-Compute	azure.com
FabricAdminClient	azsadmin-ComputeOperationResults	azure.com
StorageManagementClient	azsadmin-containers	azure.com
SubscriptionsManagementClient	azsadmin-DirectoryTenant	azure.com
ComputeDiskAdminManagementClient	azsadmin-DiskMigrationJobs	azure.com
FabricAdminClient	azsadmin-Drive	azure.com
FabricAdminClient	azsadmin-EdgeGateway	azure.com
FabricAdminClient	azsadmin-EdgeGatewayPool	azure.com
FabricAdminClient	azsadmin-Fabric	azure.com
FabricAdminClient	azsadmin-FabricLocation	azure.com
StorageManagementClient	azsadmin-farms	azure.com
FabricAdminClient	azsadmin-FileShare	azure.com
GalleryManagementClient	azsadmin-GalleryItem	azure.com
FabricAdminClient	azsadmin-InfraRole	azure.com
FabricAdminClient	azsadmin-InfraRoleInstance	azure.com
InfrastructureInsightsManagementClient	azsadmin-InfrastructureInsights	azure.com
FabricAdminClient	azsadmin-IpPool	azure.com
KeyVaultManagementClient	azsadmin-KeyVault	azure.com
NetworkAdminManagementClient	azsadmin-LoadBalancers	azure.com
FabricAdminClient	azsadmin-LogicalNetwork	azure.com
FabricAdminClient	azsadmin-LogicalSubnet	azure.com
FabricAdminClient	azsadmin-MacAddressPool	azure.com
SubscriptionsManagementClient	azsadmin-Manifest	azure.com
NetworkAdminManagementClient	azsadmin-Network	azure.com
FabricAdminClient	azsadmin-NetworkOperationResults	azure.com
SubscriptionsManagementClient	azsadmin-Offer	azure.com
SubscriptionsManagementClient	azsadmin-OfferDelegation	azure.com
FabricAdminClient	azsadmin-Operations	azure.com
SubscriptionsManagementClient	azsadmin-Plan	azure.com
Compute Admin Client	azsadmin-PlatformImages	azure.com
NetworkAdminManagementClient	azsadmin-PublicIpAddresses	azure.com
SubscriptionsManagementClient	azsadmin-Quota	azure.com
Compute Admin Client	azsadmin-Quotas	azure.com
InfrastructureInsightsManagementClient	azsadmin-RegionHealth	azure.com
InfrastructureInsightsManagementClient	azsadmin-ResourceHealth	azure.com
FabricAdminClient	azsadmin-ScaleUnit	azure.com
FabricAdminClient	azsadmin-ScaleUnitNode	azure.com
InfrastructureInsightsManagementClient	azsadmin-ServiceHealth	azure.com
FabricAdminClient	azsadmin-SlbMuxInstance	azure.com
StorageManagementClient	azsadmin-storage	azure.com
StorageManagementClient	azsadmin-storageaccounts	azure.com
FabricAdminClient	azsadmin-StorageOperationResults	azure.com
FabricAdminClient	azsadmin-StoragePool	azure.com
FabricAdminClient	azsadmin-StorageSubSystem	azure.com
FabricAdminClient	azsadmin-StorageSystem	azure.com
SubscriptionsManagementClient	azsadmin-Subscriptions	azure.com
UpdateAdminClient	azsadmin-Update	azure.com
UpdateAdminClient	azsadmin-UpdateLocations	azure.com
NetworkAdminManagementClient	azsadmin-VirtualNetworks	azure.com
Compute Admin Client	azsadmin-VMExtensions	azure.com
FabricAdminClient	azsadmin-Volume	azure.com
Azure Stack Azure Bridge Client	azurestack-AzureStack	azure.com
AzureStack Azure Bridge Client	azurestack-CustomerSubscription	azure.com
AzureStack Azure Bridge Client	azurestack-Product	azure.com
Azure Stack Azure Bridge Client	azurestack-Registration	azure.com
BlockchainManagementClient	blockchain	azure.com
BlueprintClient	blueprint-assignmentOperation	azure.com
BlueprintClient	blueprint-blueprintAssignment	azure.com
BlueprintClient	blueprint-blueprintDefinition	azure.com
Azure Bot Service	botservice	azure.com
Azure CDN WebApplicationFirewallManagement	cdn-cdnwebapplicationfirewall	azure.com
CognitiveServicesManagementClient	cognitiveservices	azure.com
Anomaly Detector Client	cognitiveservices-AnomalyDetector	azure.com
Anomaly Finder Client	cognitiveservices-AnomalyFinder	azure.com
Computer Vision	cognitiveservices-ComputerVision	azure.com
Form Recognizer Client	cognitiveservices-FormRecognizer	azure.com
Ink Recognizer Client	cognitiveservices-InkRecognizer	azure.com
Language Understanding Intelligent Service (LU...	cognitiveservices-LUIS-Runtime	azure.com
UsageManagementClient	commerce	azure.com
ContainerServiceClient	compute-containerService	azure.com
DiskResourceProviderClient	compute-disk	azure.com
SharedImageGalleryServiceClient	compute-gallery	azure.com
RunCommandsClient	compute-runCommands	azure.com
ComputeManagementClient	compute-skus	azure.com

ComputeManagementConvenienceClient	compute-swagger	azure.com
ContainerRegistryManagementClient	containerregistry	azure.com
ContainerRegistryManagementClient	containerregistry-containerregistry_build	azure.com
ContainerRegistryManagementClient	containerregistry-containerregistry_scopemap	azure.com
ContainerServiceClient	containerservice-containerService	azure.com
ContainerServiceClient	containerservice-location	azure.com
ContainerServiceClient	containerservices-containerService	azure.com
ContainerServiceClient	containerservices-location	azure.com
DatabricksClient	databricks	azure.com
Azure Data Catalog Resource Provider	datacatalog	azure.com
DataLakeAnalyticsAccountManagementClient	datalake-analytics-account	azure.com
DataLakeStoreAccountManagementClient	datalake-store-account	azure.com
iotDpsClient	deviceprovisioningservices-iotdps	azure.com
Domain Services Resource Provider	domainservices	azure.com
Azure Enterprise Knowledge Graph Service	EnterpriseKnowledgeGraph-EnterpriseKnowledgeGr...	azure.com
EventHub2018PreviewManagementClient	eventhub-EventHub-preview	azure.com
FrontDoorManagementClient	frontdoor	azure.com
WebApplicationFirewallManagement	frontdoor-webapplicationfirewall	azure.com
GuestConfiguration	guestconfiguration	azure.com
GuestConfiguration	guestconfiguration-guestconfiguration_NotImple...	azure.com
HanaManagementClient	hanaonazure	azure.com
HDInsightManagementClient	hdinsight-applications	azure.com
HDInsightManagementClient	hdinsight-capabilities	azure.com
HDInsightManagementClient	hdinsight-cluster	azure.com
HDInsightManagementClient	hdinsight-configurations	azure.com
HDInsightManagementClient	hdinsight-extensions	azure.com
HDInsightManagementClient	hdinsight-locations	azure.com
HDInsightManagementClient	hdinsight-operations	azure.com
HDInsightManagementClient	hdinsight-scriptActions	azure.com
HealthcareApisClient	healthcareapis-healthcare-apis	azure.com
VirtualMachineImageTemplate	imagebuilder	azure.com
InstanceMetadataClient	imds	azure.com
IoTSpacesClient	iotspaces	azure.com
KeyVaultManagementClient	keyvault-providers	azure.com
KeyVaultManagementClient	keyvault-secrets	azure.com
Azure Location Based Services Resource Provider	locationbasedservices	azure.com
Azure ML Commitment Plans Management Client	machinelearning-commitmentPlans	azure.com
Machine Learning Compute Management Client	machinelearningcompute-machineLearningCompute	azure.com
ML Team Account Management Client	machinelearningexperimentation-machineLearning...	azure.com
HyperDrive	machinelearningservices-hyperdrive	azure.com
Azure Machine Learning Workspaces	machinelearningservices-machineLearningServices	azure.com
Azure Machine Learning Model Management Service	machinelearningservices-modelManagement	azure.com
Machine Learning Workspaces Management Client	machinelearning-workspaces	azure.com
MaintenanceManagementClient	maintenance-Maintenance	azure.com
ManagedNetworkManagementClient	managednetwork-managedNetwork	azure.com
ManagedServicesClient	managedservices	azure.com
ACE Provisioning ManagementPartner	managementpartner-ManagementPartner	azure.com
Azure Maps Resource Provider	maps-maps-management	azure.com
MarketplaceOrdering.Agreements	marketplaceordering-Agreements	azure.com
MediaServicesManagementClient	mediaservices-media	azure.com
Mixed Reality	mixedreality	azure.com
Mixed Reality	mixedreality-proxy	azure.com
Mixed Reality	mixedreality-remote-rendering	azure.com
Mixed Reality	mixedreality-spatial-anchors	azure.com
Azure Action Groups	monitor-actionGroups_API	azure.com
Azure Activity Log Alerts	monitor-activityLogAlerts_API	azure.com
MonitorManagementClient	monitor-activityLogs_API	azure.com
MonitorManagementClient	monitor-alertRules_API	azure.com
MonitorManagementClient	monitor-alertRulesIncidents_API	azure.com
MonitorManagementClient	monitor-autoscale_API	azure.com
MonitorManagementClient	monitor-baseline_API	azure.com
MonitorManagementClient	monitor-calculateBaseline_API	azure.com
MonitorManagementClient	monitor-diagnosticsSettings_API	azure.com
MonitorManagementClient	monitor-diagnosticsSettingsCategories_API	azure.com
MonitorManagementClient	monitor-eventCategories_API	azure.com
Guest Diagnostic Settings	monitor-guestDiagnosticSettings_API	azure.com
Guest Diagnostic Settings Association	monitor-guestDiagnosticSettingsAssociation_API	azure.com
MonitorManagementClient	monitor-logProfiles_API	azure.com
MonitorManagementClient	monitor-metricAlert_API	azure.com
MonitorManagementClient	monitor-metricBaselines_API	azure.com
MonitorManagementClient	monitor-metricDefinitions_API	azure.com
MonitorManagementClient	monitor-metricNamespaces_API	azure.com
MonitorManagementClient	monitor-metrics_API	azure.com
Azure Metrics	monitor-metricsCreate_API	azure.com
MonitorManagementClient	monitor-operations_API	azure.com
Microsoft Insights	monitor-scheduledQueryRule_API	azure.com
MonitorManagementClient	monitor-serviceDiagnosticsSettings_API	azure.com
MonitorManagementClient	monitor-tenantActivityLogs_API	azure.com
VM Insights Onboarding	monitor-vmInsightsOnboarding_API	azure.com
Microsoft NetApp	netapp	azure.com
NetworkManagementClient	network-applicationSecurityGroup	azure.com
NetworkManagementClient	network-azureFirewall	azure.com
NetworkManagementClient	network-bastionHost	azure.com
NetworkManagementClient	network-ddosProtectionPlan	azure.com
NetworkManagementClient	network-expressRouteCircuit	azure.com
ExpressRouteCrossConnection REST APIs	network-expressRouteCrossConnection	azure.com
NetworkManagementClient	network-expressRouteGateway	azure.com
NetworkManagementClient	network-expressRoutePort	azure.com
NetworkManagementClient	network-firewallPolicy	azure.com
NetworkManagementClient	network-natGateway	azure.com
NetworkManagementClient	network-networkWatcher	azure.com
NetworkManagementClient	network-networkWatcherConnectionMonitorV1	azure.com
NetworkManagementClient	network-publicIpPrefix	azure.com
NetworkManagementClient	network-virtualRouter	azure.com
Azure Log Analytics	operationalinsights-Clusters	azure.com
Azure Log Analytics	operationalinsights-OperationalInsights	azure.com

Azure Log Analytics - Operations Management	operationsmanagement-OperationsManagement	azure.com
PeeringManagementClient	peering	azure.com
PolicyTrackedResourcesClient	policyinsights-policyTrackedResources	azure.com
PowerBIDedicated	powerbidedicated	azure.com
Power BI Embedded Management Client	powerbiembedded	azure.com
iotDpsClient	provisioningservices-iotdps	azure.com
RecoveryServicesBackupClient	recoveryservices-backup	azure.com
RecoveryServicesBackupClient	recoveryservicesbackup-bms	azure.com
RecoveryServicesBackupClient	recoveryservicesbackup-jobs	azure.com
RecoveryServicesBackupClient	recoveryservicesbackup-operations	azure.com
RecoveryServicesBackupClient	recoveryservicesbackup-registeredIdentities	azure.com
RecoveryServicesClient	recoveryservices-registeredidentities	azure.com
RecoveryServicesClient	recoveryservices-replicationusages	azure.com
RecoveryServicesClient	recoveryservices-vaults	azure.com
RecoveryServicesClient	recoveryservices-vaultusages	azure.com
RedisManagementClient	redis	azure.com
Azure Reservation	reservations	azure.com
Azure Resource Graph	resourcegraph	azure.com
Azure Resource Graph Query	resourcegraph-graphquery	azure.com
Microsoft.ResourceHealth	resourcehealth	azure.com
Microsoft.ResourceHealth	resourcehealth-ResourceHealth	azure.com
FeatureClient	resources-features	azure.com
ManagementLinkClient	resources-links	azure.com
ManagementLockClient	resources-locks	azure.com
ApplicationClient	resources-managedapplications	azure.com
PolicyClient	resources-policy	azure.com
PolicyClient	resources-policyAssignments	azure.com
PolicyClient	resources-policyDefinitions	azure.com
PolicyClient	resources-policySetDefinitions	azure.com
SubscriptionClient	resources-subscriptions	azure.com
SchedulerManagementClient	scheduler	azure.com
Security Center	security	azure.com
Security Center	security-adaptiveNetworkHardenings	azure.com
Security Center	security-advancedThreatProtectionSettings	azure.com
Security Center	security-alerts	azure.com
Security Center	security-allowedConnections	azure.com
Security Center	security-applicationWhitelistings	azure.com
Security Center	security-autoProvisioningSettings	azure.com
Security Center	security-complianceResults	azure.com
Security Center	security-compliances	azure.com
Security Center	security-deviceSecurityGroups	azure.com
Security Center	security-discoveredSecuritySolutions	azure.com
Security Center	security-externalSecuritySolutions	azure.com
Security Center	security-informationProtectionPolicies	azure.com
Security Center	security-iotSecuritySolutionAnalytics	azure.com
Security Center	security-iotSecuritySolutions	azure.com
Security Center	security-jitNetworkAccessPolicies	azure.com
Security Center	security-pricings	azure.com
Security Center	security-regulatoryCompliance	azure.com
Security Center	security-securityContacts	azure.com
Security Center	security-subAssessments	azure.com
Security Center	security-tasks	azure.com
Security Center	security-topologies	azure.com
Security Center	security-workspaceSettings	azure.com
MicrosoftSerialConsoleClient	serialconsole	azure.com
ServerManagement	servermanagement	azure.com
ServiceBusManagementClient	servicebus-servicebus-preview	azure.com
ServiceFabricManagementClient	servicefabric-cluster	azure.com
SignalRManagementClient	signalr	azure.com
Software Plan RP	softwareplan	azure.com
SqlManagementClient	sql-advisors	azure.com
Azure SQL Database Backup Long Term Retention ...	sql-backupLongTermRetentionPolicies	azure.com
Azure SQL Server Backup Long Term Retention Vault	sql-backupLongTermRetentionVaults	azure.com
Azure SQL Database Backup	sql-backups	azure.com
SqlManagementClient	sql-blobAuditing	azure.com
SqlManagementClient	sql-blobAuditingPolicies	azure.com
SqlManagementClient	sql-cancelOperations	azure.com
SqlManagementClient	sql-cancelPoolOperations	azure.com
SqlManagementClient	sql-capabilities	azure.com
Azure SQL Database	sql-checkNameAvailability	azure.com
Azure SQL Server API spec	sql-connectionPolicies	azure.com
SqlManagementClient	sql-databaseAutomaticTuning	azure.com
SqlManagementClient	sql-DatabaseSchema	azure.com
SqlManagementClient	sql-DatabaseSecurityAlertPolicies	azure.com
SqlManagementClient	sql-databaseVulnerabilityAssessmentBaselines	azure.com
SqlManagementClient	sql-databaseVulnerabilityAssessments	azure.com
SqlManagementClient	sql-databaseVulnerabilityAssessmentScans	azure.com
Azure SQL Database Datamasking Policies and Rules	sql-dataMasking	azure.com
SqlManagementClient	sql-dataWarehouseUserActivities	azure.com
Azure SQL Database	sql-deprecated	azure.com
Azure SQL Database disaster recovery configura...	sql-disasterRecoveryConfigurations	azure.com
SqlManagementClient	sql-encryptionProtectors	azure.com
SqlManagementClient	sql-FailoverDatabases	azure.com
SqlManagementClient	sql-FailoverElasticPools	azure.com
SqlManagementClient	sql-failoverGroups	azure.com
SqlManagementClient	sql-firewallRules	azure.com
Azure SQL Database	sql-geoBackupPolicies	azure.com
Azure SQL Database Import/Export spec	sql-importExport	azure.com
SqlManagementClient	sql-instanceFailoverGroups	azure.com
SqlManagementClient	sql-longTermRetention	azure.com
SqlManagementClient	sql-ManagedBackupShortTermRetention	azure.com
SqlManagementClient	sql-ManagedDatabaseSchema	azure.com
SqlManagementClient	sql-ManagedDatabaseSecurityAlertPolicies	azure.com
SqlManagementClient	sql-managedDatabaseSensitivityLabels	azure.com
SqlManagementClient	sql-managedDatabaseVulnerabilityAssesmentRuleB...	azure.com
SqlManagementClient	sql-managedDatabaseVulnerabilityAssessments	azure.com

SqlManagementClient	sql-managedDatabaseVulnerabilityAssessmentScans	azure.com
SqlManagementClient	sql-managedInstanceAdministrators	azure.com
SqlManagementClient	sql-ManagedInstanceEncryptionProtectors	azure.com
SqlManagementClient	sql-ManagedInstanceKeys	azure.com
SqlManagementClient	sql-ManagedInstanceTdeCertificates	azure.com
SqlManagementClient	sql-ManagedInstanceVulnerabilityAssessments	azure.com
SqlManagementClient	sql-ManagedRestorableDroppedDatabaseBackupShor...	azure.com
SqlManagementClient	sql-ManagedServerSecurityAlertPolicy	azure.com
Azure SQL Database	sql-metrics	azure.com
SqlManagementClient	sql-operations	azure.com
SqlManagementClient	sql-PrivateEndpointConnections	azure.com
SqlManagementClient	sql-PrivateLinkResources	azure.com
Azure SQL Database	sql-queries	azure.com
Azure SQL Database	sql-recommendedElasticPools	azure.com
Azure SQL Database	sql-recommendedElasticPoolsDecoupled	azure.com
SqlManagementClient	sql-recoverableManagedDatabases	azure.com
SqlManagementClient	sql-renameDatabase	azure.com
Azure SQL Database replication links	sql-replicationLinks	azure.com
SqlManagementClient	sql-restorableDroppedManagedDatabases	azure.com
SqlManagementClient	sql-restorePoints	azure.com
SqlManagementClient	sql-SensitivityLabels	azure.com
SqlManagementClient	sql-serverAutomaticTuning	azure.com
Azure SQL Database API spec	sql-serverAzureADAdministrators	azure.com
Azure SQL Database	sql-serverCommunicationLinks	azure.com
SqlManagementClient	sql-serverDnsAliases	azure.com
SqlManagementClient	sql-serverKeys	azure.com
SqlManagementClient	sql-serverSecurityAlertPolicies	azure.com
SqlManagementClient	sql-ServerVulnerabilityAssessments	azure.com
Azure SQL Database	sql-serviceObjectives	azure.com
SqlManagementClient	sql-shortTermRetentionPolicies	azure.com
Azure SQL Database	sql-sql.core	azure.com
SqlManagementClient	sql-syncAgents	azure.com
SqlManagementClient	sql-syncGroups	azure.com
SqlManagementClient	sql-syncMembers	azure.com
SqlManagementClient	sql-tableAuditing	azure.com
SqlManagementClient	sql-TdeCertificates	azure.com
SqlManagementClient	sql-usages	azure.com
SqlManagementClient	sql-virtualclusters	azure.com
SqlManagementClient	sql-virtualNetworkRules	azure.com
StorageManagementClient	storage	azure.com
StorageManagementClient	storage-blob	azure.com
Azure Data Lake Storage	storage-DataLakeStorage	azure.com
StorageImportExport	storageimportexport	azure.com
StorageManagementClient	storage-managementpolicy	azure.com
StreamAnalyticsManagementClient	streamanalytics-functions	azure.com
StreamAnalyticsManagementClient	streamanalytics-inputs	azure.com
StreamAnalyticsManagementClient	streamanalytics-outputs	azure.com
StreamAnalyticsManagementClient	streamanalytics-streamingjobs	azure.com
StreamAnalyticsManagementClient	streamanalytics-subscriptions	azure.com
StreamAnalyticsManagementClient	streamanalytics-transformations	azure.com
SubscriptionClient	subscription-operations	azure.com
SubscriptionDefinitionsClient	subscription-subscriptionDefinitions	azure.com
SubscriptionClient	subscription-subscriptions	azure.com
Visual Studio Resource Provider Client	visualstudio-Csm	azure.com
Visual Studio Projects Resource Provider Client	visualstudio-PipelineTemplates	azure.com
Visual Studio Projects Resource Provider Client	visualstudio-Projects	azure.com
AppServiceCertificateOrders API Client	web-AppServiceCertificateOrders	azure.com
CertificateRegistrationProvider API Client	web-CertificateRegistrationProvider	azure.com
Certificates API Client	web-Certificates	azure.com
DeletedWebApps API Client	web-DeletedWebApps	azure.com
DomainRegistrationProvider API Client	web-DomainRegistrationProvider	azure.com
Domains API Client	web-Domains	azure.com
LogicAppsManagementClient	web-logicAppsManagementClient	azure.com
Recommendations API Client	web-Recommendations	azure.com
ResourceHealthMetadata API Client	web-ResourceHealthMetadata	azure.com
API Client	web-ResourceProvider	azure.com
TopLevelDomains API Client	web-TopLevelDomains	azure.com
DeviceServices	windowsiot-WindowsIotServices	azure.com
Beanstream Payments		beanstream.com
Betfair: Exchange Streaming		betfair.com
Bhagavad Gita		bhagavadgita.io
Browshot		browshot.com
Bufferapp		bufferapp.com
BulkSMS JSON		bulksms.com
Call Control		callcontrol.com
CarbonDoomsDay		carbondoomsday.com
CircleCI		circleci.com
GoToMeeting	gotomeeting	citrixonline.com
SCIM	scim	citrixonline.com
ocrapi	ocr	cloudmersive.com
CodeScan		code-scan.com
Crossbrowsertesting.com Screenshot Comparisons		crossbrowsertesting.com
D7SMS		d7networks.com
Open Skills		dataatwork.org
Deep Art Effects		deeparteffects.com
DLx		digitallinguistics.io
Browse	buy-browse	ebay.com
Fulfillment	sell-fulfillment	ebay.com
elmah.io		elmah.io
U.S. EPA Enforcement and Compliance History On...	sdw	epa.gov
EVEMarketer Marketstat		evemarketer.com
GoToTraining	gototraining	getgo.com
Giphy		giphy.com
Gisgraphy webservices		gisgraphy.com
Abusive Experience Report	abusiveexperiencereport	googleapis.com
Ad Exchange Seller	adexchangeseller	googleapis.com

Ad Experience Report	adexperiencereport	googleapis.com
Admin Reports	admin	googleapis.com
AdSense Host	adsensehost	googleapis.com
Drive Activity	appsactivity	googleapis.com
Google App State	appstate	googleapis.com
BigQuery Reservation	bigqueryreservation	googleapis.com
Hangouts Chat	chat	googleapis.com
Cloud User Accounts	clouduseraccounts	googleapis.com
Cloud Composer	composer	googleapis.com
CustomSearch	customsearch	googleapis.com
Google Cloud Deployment Manager	deploymentmanager	googleapis.com
Google Cloud DNS	dns	googleapis.com
Domains RDAP	domainsrdap	googleapis.com
DoubleClick Bid Manager	doubleclickbidmanager	googleapis.com
Search Ads 360	doubleclicksearch	googleapis.com
Firebase Remote Config	firebaseremoteconfig	googleapis.com
Fusion Tables	fusiontables	googleapis.com
Google Play Game Services Publishing	gamesConfiguration	googleapis.com
Groups Migration	groupsmigration	googleapis.com
Groups Settings	groupssettings	googleapis.com
Google Identity Toolkit	identitytoolkit	googleapis.com
Licensing	licensing	googleapis.com
Manufacturer Center	manufacturers	googleapis.com
Google Mirror	mirror	googleapis.com
Google OAuth2	oauth2	googleapis.com
Google Play Custom App Publishing	playcustomapp	googleapis.com
Google Play Movies Partner	playmoviespartner	googleapis.com
Google+	plus	googleapis.com
Google+ Domains	plusDomains	googleapis.com
Poly	poly	googleapis.com
Prediction	prediction	googleapis.com
QPX Express	qpxExpress	googleapis.com
Google Compute Engine Instance Group Manager	replicapool	googleapis.com
Google Compute Engine Instance Group Updater	replicapoolupdater	googleapis.com
Enterprise Apps Reseller	reseller	googleapis.com
Google Compute Engine Instance Groups	resourceviews	googleapis.com
Google Site Verification	siteVerification	googleapis.com
Google Spectrum Database	spectrum	googleapis.com
TaskQueue	taskqueue	googleapis.com
Google Cloud Translation	translate	googleapis.com
URL Shortener	urlshortener	googleapis.com
Google Fonts Developer	webfonts	googleapis.com
Search Console	webmasters	googleapis.com
YouTube Analytics	youtubeAnalytics	googleapis.com
YouTube Reporting	youtubereporting	googleapis.com
Route Optimization		graphhopper.com
Greenwire Public		greenpeace.org
Positioning	positioning	here.com
Active Documentation for /v1		idtbeyond.com
import.io	run	import.io
import.io	schedule	import.io
Mailsquad		inboxroute.com
Infermedica		infermedica.com
API iSendPro		isendpro.com
Jokes One		jokes.one
Just Eat UK		just-eat.co.uk
KeyServ		keyserv.solutions
koomalooma Partner		koomalooma.com
Deed	deed	landregistry.gov.uk
AgentOS Api V3, diary	diary	letmc.com
AgentOS Api V3, maintenance	maintenance	letmc.com
link.fish		link.fish
LotaData		lotadata.com
Lumminary		lumminary.com
Lyft		lyft.com
Tradeworks		magick.nu
Medium.com		medium.com
Dealer	dealer	mercedes-benz.com
Remote Diagnostic Support	diagnostics	mercedes-benz.com
Vehicle Image	image	mercedes-benz.com
Miataru		miataru.com
Computer Vision Client	cognitiveservices-Ocr	microsoft.com
Custom Vision Prediction Client	cognitiveservices-Prediction	microsoft.com
Spell Check Client	cognitiveservices-SpellCheck	microsoft.com
n-Auth		n-auth.com
Netatmo		netatmo.net
Labs64 NetLicensing RESTful API Test Center		netlicensing.io
NPR Authorization Service	authorization	npr.org
NPR Identity Service	identity	npr.org
NPR Listening Service	listening	npr.org
NPR Sponsorship Service	sponsorship	npr.org
NPR Station Finder Service	station-finder	npr.org
Transportation Laws and Incentives	transportation-incentives-laws	nrel.gov
Books	books_api	nytimes.com
Community	community	nytimes.com
Geographic	geo_api	nytimes.com
Most Popular	most_popular_api	nytimes.com
Movie Reviews	movie_reviews	nytimes.com
obono RKSV		obono.at
OMDb		omdbapi.com
OpenALPR Cloud		openalpr.com
OpenAPI space		openapi.space
OpenCage Geocoder		opencagedata.com
OpenFinTech.io		openfintech.io
groov Public	groov	opto22.com
paccurate.io		paccurate.io

PasswordUtility.Web		passwordutility.net
GOV.UK Pay	payments	payments.service.gov.uk
Authentication	authentication	personio.de
Postmark Account-level	account	postmarkapp.com
Qualpay Payment Gateway		qualpay.com
They Said So Quotes		quotes.rest
Request Baskets		rbaskets.in
Receptive		receptive.io
CompanyAPI		roaring.io
Seldon		rummblelabs.com
Runscope		runscope.com
SelectPdf HTML To PDF		selectpdf.com
setlist.fm		setlist.fm
SimplyRETS		simplyrets.com
SlideRoom API V2		slideroom.com
SpectroCoin Merchant		spectrocoin.com
Spinitron v2		spinitron.com
StatSocial Platform		statsocial.com
Storecove		storecove.com
SwaggerHub Registry		swaggerhub.com
Swagger Generator	generator	swagger.io
SYNQ Video		synq.fm
Taxrates.io		taxrates.io
api.text2data.org		text2data.org
Airports API v2		transavia.com
TransitFeeds		transitfeeds.com
TVmaze user		tvmaze.com
Gateway		tyk.com
VAT API		vatapi.com
VisibleThread		visiblethread.com
Voodoo Manufacturing 3D Print		voodoomfg.com
The Water Linked Underwater GPS		waterlinked.com
Wavecell.Sms.Api		wavecell.com
WeGA		weber-gesamtausgabe.de
Accounts	accounts	whapi.com
Bets	bets	whapi.com
Sessions	sessions	whapi.com
WINSMS		winsms.co.za
Word Associations		wordassociations.net
Zalando Shop		zalando.com
Zappiti Player		zappiti.com